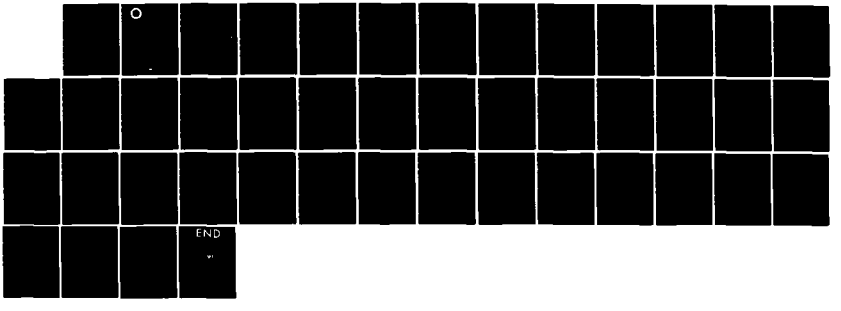
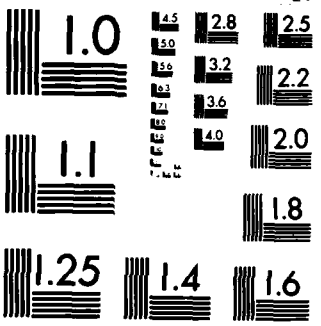
AD-A155 720 MODEL BUILDING AND PRACTICAL ASPECTS OF NONLINEAR PROGRAMMING(U) STANFORD UNIV CA SYSTEMS OPTIMIZATION LAB P E GILL ET AL. MAR 85 SOL-85-2 ARO-21592.2-MA 1/1

UNCLASSIFIED N00014-75-C-0267 F/G 12/1 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ARO 21072.2-MA

Systems
Optimization
Laboratory

MODEL BUILDING AND PRACTICAL ASPECTS OF NONLINEAR PROGRAMMING†

by

Philip E. Gill, Walter Murray,
Michael A. Saunders and Margaret H. Wright

TECHNICAL REPORT SOL 85-2

March 1985

Department of Operations Research
Stanford University
Stanford, CA 94305

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305

MODEL BUILDING AND PRACTICAL ASPECTS OF NONLINEAR PROGRAMMING†

by

Philip E. Gill, Walter Murray,
Michael A. Saunders and Margaret H. Wright

TECHNICAL REPORT SOL 85-2

March 1985

Research and reproduction of this report were partially supported by National Science Foundation Grants MCS-7926009 and ECS-8312142; U.S. Department of Energy Contract DE-AM03-76F00326, PA# DE-AT03-76ER72018; Office of Naval Research Contract N00014-75-C-0267; and U.S. Army Research Office Contract DAAG29-84-K-0156.

Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do **NOT** necessarily reflect the views of the above sponsors.

---

†Presented as an invited paper at the NATO Advanced Study Institute on "Computational Mathematical Programming", Bad Windsheim, July 23-August 2, 1984.

1. Aspects of modelling that affect optimization

1.1. Introduction. This survey paper has two main purposes: to summarize (briefly) certain aspects of modelling that influence the performance of optimization algorithms, and to describe recent advances in methods for nonlinear programming that influence the solution of practical problems. These two themes are not unconnected. A well constructed mathematical model should be such that the bad effects of ill-conditioning, degeneracy and inconsistent constraints are minimized. Ironically, the purpose of good software is to deal effectively with precisely these problems. Therefore it is not surprising that much of the insight necessary to construct a well-posed mathematical model is pertinent to the formulation of robust algorithms.

The principal problem of concern will be the *nonlinear programming problem*:

NP
$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad F(x)$$
$$\text{subject to} \quad \ell \leq \begin{Bmatrix} x \\ A_L x \\ c(x) \end{Bmatrix} \leq u,$$

where $F(x)$ (the *objective function*) is a smooth nonlinear function, A_L is a constant matrix of constraints, and $c(x)$ is a vector of smooth nonlinear constraint functions. The objective function F and the constraint functions taken together comprise the *problem functions*. Unless otherwise stated, the problem functions will be assumed to be at least twice-continuously differentiable. (Methods that require this degree of smoothness will usually work if there are isolated discontinuities away from the solution). We shall use $g(x)$ to denote the gradient of $F(x)$, and $a_i(x)$ the gradient of $c_i(x)$. The solution of NP will be denoted by x^*.

We shall begin with a statement of ten "modelling principles" that may help to make the resulting problem NP more susceptible to existing nonlinear programming software. This is followed by a review of the practical aspects of quasi-Newton sequential quadratic programming (SQP) methods for nonlinear programming. We conclude by presenting examples of the application of SQP methods to some illustrative optimization problems.

1.2. Some basic modelling principles. Our observations of practical optimization problems have indicated that, even with the best available software, the efficient optimization of a model can be critically dependent on certain properties of the formulation. It is often the case that the formulator of the model must make numerous decisions that do not affect the accuracy to which the model reflects the real world, yet are crucial to whether the model is amenable to solution by an optimization algorithm.

Our experience with the role of modelling in numerical optimization will be summarized by a list of ten "modelling principles". These principles may serve as a guide for those who have little knowledge of the intricacies and potential pitfalls of modern optimization codes. They have been derived from our own experiences with real problems.

Of course, the nature of possible models varies so much that it is impossible to treat all relevant aspects of modelling. The main thesis of these principles is that developers of models

should consider in the initial stages the ultimate need to solve an optimization problem, since it is unlikely that optimization software will ever reach the state wherein a general routine can be used with impunity for all problems.

For additional material on aspects of modelling that influence the performance of optimization methods, the reader is referred to Gill, Murray and Wright (1981).

BASIC MODELLING PRINCIPLES

I. Formulate a simple model first and add features in conjunction with running the optimization.

A model to be optimized should be developed by striking a reasonable balance between the aims of improved accuracy in the model (which usually implies added complexity in the formulation) and increased ease of optimization. This might be achieved by invoking an optimization procedure on successively more complicated versions of the model, in a form of "stepwise" refinement. Thus, the effects of each refinement in the model on the optimization process can be monitored, and fundamental difficulties can be discovered much more quickly than if no optimization were applied until the model was essentially complete. This is especially important when dealing with models that contain many interconnected subsystems, each requiring extensive calculation.

II. Attempt to use smooth problem functions.

Probably the most fundamental property of the problem functions with respect to ease of optimization is *differentiability*, which is important because algorithms are based on using available information about a function at one point to deduce its behavior at other points. If the problem functions are twice-continuously differentiable, say, the ability of an algorithm to locate the solution is greatly enhanced compared to the case when the problem functions are non-differentiable. Therefore, most optimization software is designed to solve smooth problems, and there is a great incentive to formulate differentiable model functions. A useful feature of methods for smooth problems is that they tend to give more information concerning the *quality* of the solution. For example, some nonlinear programming methods can be shown to exhibit a superlinear rate of convergence in the neighborhood of a local minimum. If a method terminates at a point for which this rate of convergence is exhibited, the user will have some confidence that the final point is close to a local minimum. (When solving a problem on a digital computer, we need to define carefully what we mean by a "smooth" problem. In reality, all software minimizes a function $fl(F(x))$, which is the floating-point representation of $F(x)$. The function $fl(F(x))$ is piecewise constant at the round-off level. If we define ϵ_A to be the absolute precision of F, i.e.,

$$|F(x) - fl(F(x))| = \epsilon_A(x),$$

then algorithms for smooth problems will work whenever changes in the variables produce changes in F that are much greater than ϵ_A. Note that the vast majority of optimization software assumes

that F is computed to full precision; i.e., it is assumed that ϵ_A is of the order of $\epsilon_M \|F(x)\|$, where ϵ_M is the relative machine precision.)

III. Avoid defining problem functions that are the result of some iterative procedure (such as the solution of a differential equation or the evaluation of an integral).

Problem functions defined by an iterative procedure are often the source of subtle discontinuities that may impede the progress of the optimization. The solution of these subproblems to full machine precision (even if possible) generally requires considerable computational effort, and thus tends to be regarded as unwarranted by the modeller, since the integral or differential equation (or whatever) is only an approximation to some more complicated real-world phenomenon.

The use of an iterative procedure to define a problem function most often occurs when the variables of the problem are *functions of a continuous parameter* (in an optimal control problem, for example). In many instances, an effective strategy for this type of problem is to discretize the problem *before* applying the optimization method. Accurate solutions to the continuous problem are then found by refining the discretization between optimizations. Such a strategy illustrates again that it is often worthwhile to interleave modelling and optimization, since the creation of an increasingly accurate discretization is in fact a modelling process.

IV. Think carefully about the nature of the constraints.

It is not always appreciated that substantial improvements in performance and robustness can result when methods exploit the different properties of simple bounds, linear constraints and nonlinear constraints. Whenever possible, the user should isolate the linear constraints from the nonlinear constraints and use software that differentiates between constraint types during the optimization. Unfortunately, some problem formats *guarantee* that a linear constraint will be treated as a nonlinear constraint. For example, in the class of *geometric programming problems* the objective and constraint functions are sums of functions of the form

$$\phi_i(x) = \alpha_i x_1^{a_{i1}} x_2^{a_{i2}} \cdots x_n^{a_{in}}, \tag{1.1}$$

where the x_j are the variables (constrained to be positive) and the a_{ij} are constants. The transformation of a linear constraint into a sum of functions of the form (1.1) unnecessarily increases the degree of difficulty of the problem.

The transformation of a problem from one form to another was often unavoidable in the past because less software was available. When algorithms for unconstrained optimization were more numerous and more effective than for constrained problems, it was common practice for simple bound constraints to be treated by a change of variable. Today, however, algorithms for problems with only simple bounds or linear constraints are comparable in efficiency to unconstrained algorithms. Therefore, it is virtually never worthwhile to transform bound-constrained problems (in fact, it is often beneficial to *add* bounds on the variables — see below), and it is rarely appropriate to alter linearly constrained problems.

Transformations can be used effectively to transform nonlinear constraints into simple bound constraints (for example, by using polar coordinates instead of cartesian coordinates to deal with range constraints of the form $l \leq \sum x_i^2 \leq u$). However, care should be taken to ensure that the transformation does not lead to a new problem that is more difficult to solve, or has additional (spurious) solutions.

V. Do not attempt to eliminate equality constraints from the problem.

Modellers often assume that since there may be no physical significance to a point at which nonlinear equality constraints are violated, such constraints should be satisfied exactly at all stages of the optimization. Accordingly, users often attempt to "eliminate" nonlinear equality constraints from the problem by the following method. The variables are partitioned into "independent" and "dependent" sets. The minimization is then performed only with respect to the independent variables, and the dependent variables are determined by "solving" the equality constraints. To be more precise, let x denote the vector of dependent variables and u the vector of independent variables. The constrained problem

$$\begin{aligned} \underset{u,x}{\text{minimize}} \quad & F(x,u) \\ \text{subject to} \quad & c(x,u) = 0, \end{aligned}$$

is solved by expressing $c(x,u) = 0$ as $x = T(u)$ for some transformation T, and then minimizing $F(T(u),u)$ with respect to u. This strategy is particularly common when there are special fast methods for solving the equations $c(x,u) = 0$.

We do not recommend this approach if the constraints have any significant degree of nonlinearity. Firstly, it is difficult to impose any simple bounds upon the dependent variables. Secondly, the resulting algorithm is of the "constraint-following" type, which will tend to be less efficient than other methods. Our experience is that the total computational effort required to solve for the dependent variables at every trial point is not usually worthwhile, compared to expending a similar amount of effort in the optimization without eliminating the variables. The user would be better advised to use a general nonlinear programming method.

VI. Distinguish between "hard" and "soft" constraints.

In many problems, the bound constraints may be classified as either "hard" or "soft". For example, hard bounds could specify the region in which the problem functions are well-defined. A soft constraint might represent a bound whose violation we are prepared to tolerate if this resulted in a large decrease in the objective function. In some cases, variables may have both a soft bound and a hard bound.

A good method of treating a soft bound of the form $x_i \geq b_i$ is to exclude it from the simple bound constraints and include it in the objective function in the form of a mild penalty term of the form $\frac{1}{2}\rho_i [x_i - b_i]_-^2$, where $[y]_-$ denotes the function $\min\{0,y\}$. Modest values of ρ_i do not lead to the severe ill-conditioning that occurs with traditional penalty methods. If the variable is also constrained by a hard bound, the constraint can be treated explicitly in the usual way.

As a general rule, the presence of simple bounds simplifies a problem by reducing dimensionality. However, if a bound has only a slight impact upon the problem (i.e., if the objective function does not change rapidly as the bound is perturbed), the degree of difficulty of the problem may be *increased* because the algorithm needs to resolve small Lagrange multipliers. If many of the bounds with small multipliers would be classified as "soft", the use of a penalty term tends to remove these constraints from the active set.

Similarly, a penalty function of the form $\frac{1}{2}\rho_i(x_i - d_i)^2$ can be used to produce a solution that is biased towards some "desired" value d.

VII. Avoid modelling near-dependent equality constraints

Constraints occur in problem formulations for a variety of reasons. Often the very nature of the variables imposes an equality constraint — for example, if the variables $\{x_i\}$ represent proportions or probabilities, this gives rise to the constraint $\sum_i x_i = 1$ (as well as non-negativity restrictions). Constraints of this type are "genuine" equalities, in the sense that the computed solution must satisfy them exactly (where "exactly" means "within working precision"). However, it is not unusual in modelling that constraints that might seem initially to be firm equality constraints should be treated instead as constraints that need not be satisfied with maximum possible accuracy. For example, this situation occurs when the underlying model is known to contain inaccuracies. In some problems, forcing constraints of this type to be equalities may cause there to be no feasible solution, or may distort the properties of the solution if the corresponding constraints are ill-conditioned.

We shall use a simple two-dimensional example to illustrate not only the the potential difficulties associated with dependent constraints, but also one simple method for dealing with them. Suppose that we require the minimum of a function of two variables subject to two independent linear equality constraints. In this case, the solution lies at the point of intersection of the constraints (the objective function has no influence on the solution). However, suppose that one of the two constraints is really a copy of the other, but that due to small errors in the modelling process, the constraints are not *exactly* dependent. Now, the two constraints are nearly parallel and the point of intersection lies at a point that may be very different from the solution of the essentially equivalent problem posed with a *single* constraint.

A clue to one method of resolution of this difficulty lies in the observation that if one or the other of the constraints were ignored during the optimization, it would be violated by only a very small quantity. Suppose that each dependent equality constraint is replaced by an *inequality* constraint with a very narrow range. For example, the linear constraint $a^T x = b$ would be replaced by

$$b - \delta \leq a^T x \leq b + \delta,$$

where δ is a small, but not negligible, positive quantity. (Exactly the same transformation can be made for a nonlinear constraint.) In the example above, the active-set strategy of the optimization

procedure will decide that a lower value of the objective function is achieved if one of the equality constraints is not satisfied exactly at the solution.

In Section 3.5 we shall use this type of range constraint in methods that are robust on problems with dependent constraints, and show that the idea of allowing certain constraints to be violated by a small quantity recurs in many practical solution techniques. This is only to be expected, since it is never possible to satisfy constraints *exactly* in finite-precision arithmetic.

VIII. Use information about the problem to scale the variables and constraints.
It is well known that the "right" scaling of variables and constraints can dramatically improve the efficiency and accuracy of optimization methods. The *scale* of a problem is the measure of the relative importance of the variables and constraints — or, equivalently, the scale of a problem is a statement of what is "large" and what is "small" in a problem.

In the absence of any other information, an algorithm will generally assume that each variable or constraint has equal weight in the optimization. For example, if a unit change in a variable produces a small change in F compared to other variables, the algorithm will tend to make a larger change to that variable. Clearly, if the variables do have equal significance, this situation is quite satisfactory. However, if the small variation in F is due to the fact that the objective function is almost independent of the variable, the problem should be *rescaled*. (Note that in this case, fixing a variable or constraint is an acceptable form of problem scaling.)

The most common form of scaling is to define new variables and constraints using a linear transformation. (For simplicity, in the following discussion we shall assume that the nonlinear constraints are equalities of the form $c_i(x) = 0$.) Suppose we define new variables $\bar{x}$ and constraints $\bar{c}$ such that

$$x = D_1\bar{x} \qquad \text{and} \qquad \bar{c} = D_2 c, \tag{1.2}$$

where D_1 and D_2 are nonsingular matrices (usually diagonal). With this scaling, the derivatives of the original and transformed objective function are related by $\nabla_x F = D_1 \nabla F$ and $\nabla_x^2 F = D_1 \nabla^2 F D_1$. Similarly, for the constraints we have $\bar{A} = D_2 A$, where A and $\bar{A}$ are the Jacobian matrices of the original and transformed constraints. When D_1 and D_2 are diagonal, an interpretation of the scaling (1.2) is that D_1 and D_2 "rank" the elements of the gradient vector and constraints so that each is of equal importance.

The important thing to bear in mind is that a badly scaled problem is essentially an *ill-conditioned* problem. Badly scaled variables lead to ill-conditioned Hessian matrices; badly scaled constraints give rise to near-singular Jacobian matrices. Within optimization routines, convergence tolerances and other criteria are necessarily based upon an implicit definition of "small" and "large", and thus variables with widely varying orders of magnitude may cause difficulties for some algorithms.

Arguably the most important rule of scaling is that the variables of the scaled problem should be of similar magnitude and of order unity in the region of interest. If typical values of all the variables are known, a problem can be transformed so that the variables are all of the same order

of magnitude. In this case, the transformation D_1 of (1.2) is a diagonal matrix whose elements are typical values of x.

An important property of a theoretical algorithm is that of invariance with respect to the scaling (1.2). A scale-invariant algorithm has the property that when it is applied to both the original problem and the transformed problem, the resulting sequences of iterates $\{x_k\}$ and $\{\bar{x}_k\}$ satisfy $x_k = D_1\bar{x}_k$ for all k. This property does not imply that it is unnecessary to choose a sensible scaling for a problem when a scale-invariant algorithm is being used. *Scale-invariance cannot be achieved in a practical implementation of an algorithm.* Not only is computer arithmetic not scale-invariant, but also it is impossible to devise a scale-invariant test that an algorithm has converged to a point which satisfies the necessary or sufficient conditions. Many algorithms treat quantities that are "sufficiently small" in magnitude as "negligible" (in effect, as zero). However, since there is no universal definition of "small", it is impossible to formulate a scale-invariant procedure for distinguishing between quantities that *should* be treated as zero, and scale-dependent quantities that *should not* be neglected. To illustrate this point, consider the application of any quasi-Newton method that is scale-invariant in the sense just described. If the algorithm were applied to the scaled and unscaled problem, we may regard the two sequences $\{x_k\}$ and $\{\bar{x}_k\}$ as belonging to two different spaces. Each value x_k in the x-space is related to a value $\bar{x}_k$ in the $\bar{x}$-space by the formula $\bar{x}_k = D_1x_k$. The optimality test of the algorithm must involve consideration of the magnitude of the gradient norm. However, since the gradient norm is tending to zero with a different value (either $\|g_k\|$ or $\|D_1g_k\|$) in each space, this optimality test cannot be made scale-invariant. Thus, the *number* of elements in each sequence may be different. Furthermore, the initial estimate of the Hessian is almost always defined as a unit matrix. If this is done in both the x- and $\bar{x}$-spaces, the scale invariance is immediately lost.

Another part of an algorithm that is unavoidably scale-dependent is the criterion for deciding when a nonlinear constraint of the form $c_i(x) = 0$ is satisfied. Given a constraint value, say $c_i(x) = 10^{-4}$, it is necessary to know something about the natural scaling of the problem in order to make a sensible decision about whether the constraint is sufficiently close to zero. The only way of overcoming this problem is to let the user decide when a constraint is sufficiently satisfied. This leads to the idea of a user-defined *feasibility tolerance* δ_i that defines what is really "small" for each constraint. For example, the part of the termination criterion that concerns the test for feasibility would require that $|c_i(x)| \leq \delta_i$ for each constraint.

IX. Try to provide as much information about the function as possible.

As a general rule, algorithms tend to be more successful and robust when more information about the problem is provided. For example, if the problem functions are *smooth*, algorithms that use first and second derivatives perform much better than algorithms that use function values only.

It is generally perceived that a second-derivative algorithm is rarely useful because the cost of computing the second derivatives may be several orders of magnitude larger than that of calculating first derivatives. This is undoubtedly true in some situations, but there are a remarkably

large number of problems for which the first and second derivatives of the problem functions may be obtained for about the same cost — for example, geometric programming problems, where the objective and constraint functions are of the form (1.1). Moreover, it is relatively straightforward to design software that will automatically differentiate the original problem functions.

Other examples of problems with cheap higher derivatives occur in exponential fitting and factorable programming (for example, see McCormick, 1983).

X. Take special care to check that the problem functions and their derivatives are programmed correctly.

Before embarking on a series of optimization runs, the user should verify that the code which defines the problem functions is correct. One obvious check is to evaluate the problem functions at a point where their values are known.

Errors in programming the function may be quite subtle in that the function value is "almost" correct. For example, the function may be accurate to less than full precision because of the inaccurate calculation of a subsidiary quantity, or the limited accuracy of data upon which the function depends. A common error on machines where numerical calculations are usually performed in double precision is to include even one single-precision constant in the calculation of the function; since some compilers do not convert such constants to double precision, half the correct figures may be lost by such a seemingly trivial error.

Incorrect calculation of derivatives is by far the most common user error. Such errors are almost never small, and thus no algorithm can perform correctly in their presence. This is why we recommend that some sort of consistency check on the derivatives be performed. The most straightforward means of checking for errors in the derivative involves comparing a finite-difference approximation with the supposedly exact value.

1.3. Some useful features of an implementation. In the following, we give some features of an "ideal" implementation that would help the user apply the modelling principles most effectively. This list is not intended to be exhaustive and we have included some items in the list that have not been referred to in the text. We do not claim that it is possible to implement every feature in all circumstances.

- The method should treat simple bounds, linear constraints and nonlinear constraints separately. Moreover, the method should effectively deal with nonlinear constraints, yet remain competitive with unconstrained or linearly constrained methods if constraint nonlinearities are not present.
- There should be the option of computing the problem functions only at points that satisfy the linear constraints.
- The software should print all the information necessary for the user to check how closely the final point satisfies the necessary and sufficient conditions for an optimum.
- The user should be able to specify a feasibility tolerance for each constraint.

- It should be possible to solve problems for which the precision of the problem functions is not necessarily close to the machine precision.
- The user should be able to scale the problem by specifying the scaling matrices D_1 and D_2 of (1.2).
- The user should have the option of checking the problem derivatives before starting the minimization.

2. Quasi-Newton methods for unconstrained minimization

Quasi-Newton methods are *iterative*, and generate a sequence $\{x_k\}$ that is intended to converge to x^*. At the k-th iteration, the new iterate is defined by

$$x_{k+1} = x_k + \alpha_k p_k, \tag{2.1}$$

where α_k is a non-negative scalar called the *step length*, and p_k is an n-vector called the *search direction*. These methods utilize the values of the problem functions and their gradients at trial iterates, but do not assume the availability of higher derivative information. (When explicit first derivatives are not available, quasi-Newton methods can be implemented using finite-difference approximations to the gradient -- see, e.g., Gill, Murray and Wright, 1981.) A typical iteration of a quasi-Newton method comprises three related parts: computation of p_k, choice of α_k, and updating the necessary matrix factorizations.

The search direction p_k in unconstrained optimization is defined by the equation $H_k p_k = -g_k$, where H_k is an approximation to the Hessian matrix of F. Many computational benefits accrue from updating a factorized form of H_k. Suppose that $H_k = R_k^T R_k$, where R_k is an upper-triangular matrix (the Cholesky factor of H_k). The search direction is then obtained by solving two triangular systems:

$$R_k^T q = -g_k \quad \text{and} \quad R_k p_k = q. \tag{2.2}$$

The main purpose of the linesearch is to force steady progress by computing a step length α_k such that $F(x_k + \alpha_k p_k)$ is "sufficiently less" than $F(x_k)$; i.e., the decrease can go to zero only as the solution is approached.

It is widely accepted today that the best quasi-Newton update is given by the *BFGS formula*:

$$H_{k+1} = H_k - \frac{1}{u_k^T s_k} u_k u_k^T + \frac{1}{y_k^T s_k} y_k y_k^T, \tag{2.3}$$

where $s_k = x_{k+1} - x_k$, $u_k = H_k s_k$ and $y_k = g_{k+1} - g_k$. If H_k is positive definite, H_{k+1} as defined by (2.3) will be positive definite if and only if the approximate curvature $y_k^T s_k$ satisfies

$$y_k^T s_k > 0 \tag{2.4}$$

(see Dennis and Moré, 1977).

Dennis and Schnabel (1981) have shown that the BFGS update (2.3) may be expressed as a *rank-one* update to R_k. Let β and γ denote the scalars $(u_k^T s_k)^{\frac{1}{2}}$ and $(y_k^T s_k)^{\frac{1}{2}}$ respectively. The BFGS update may then be written as $H_{k+1} = \bar{R}_{k+1}^T \bar{R}_{k+1}$, where

$$\bar{R}_{k+1} = R_k + vw^T, \quad \text{with} \quad v = \frac{1}{\beta} R_k s_k, \quad w = \frac{1}{\gamma} y_k - \frac{1}{\beta} u_k.$$

The matrix $\bar{R}_k$, which is not upper-triangular, may be restored to upper-triangular form by finding an orthogonal matrix P such that

$$P\bar{R}_{k+1} = R_{k+1},$$

where R_{k+1} is upper triangular. Then $H_{k+1} = \bar{R}_{k+1}^T \bar{R}_{k+1} = \bar{R}_{k+1}^T P^T P \bar{R}_{k+1} = R_{k+1}^T R_{k+1}$, as required. A suitable matrix P can be constructed from two sweeps of plane rotations; for more details, see Gill *et al.* (1974).

Since p_k satisfies $H_k p_k = -g_k$, two matrix-vector multiplications may be avoided in the implementation of the BFGS update (2.3). Substituting from (2.2), we obtain

$$v = \frac{1}{\sigma} q \quad \text{and} \quad w = \frac{1}{\gamma} y_k + \frac{1}{\sigma} g_k,$$

where $\sigma = |g_k^T p_k|^{\frac{1}{2}}$ and $\gamma = (y_k^T s_k)^{\frac{1}{2}}$.

Representation of H_k by its Cholesky factors avoids a serious problem that would otherwise arise in quasi-Newton methods: the loss of positive-definiteness through rounding errors. With exact arithmetic, satisfaction of (2.4) should ensure that the BFGS update generates strictly positive-definite Hessian approximations. However, in practice the formula (2.3) can lead to a singular or indefinite matrix H_{k+1}. When H_k is represented by its Cholesky factorization and updates are performed directly to the factorization, every H_k will be *numerically* positive definite.

Maintenance of positive-definiteness is considered to be a crucial element in the success of quasi-Newton methods in unconstrained optimization (see Dennis and Moré, 1977). In this context, the linesearch can also have the important function of guaranteeing that condition (2.4) holds at every iteration. In particular, methods of safeguarded polynomial interpolation (see, e.g., Brent, 1973; Gill, Murray and Wright, 1981) can find a value of α_k that satisfies

$$F(x_k) - F(x_k + \alpha_k p_k) \geq -\mu \alpha_k g_k^T p_k, \tag{2.5a}$$

$$|g(x_k + \alpha_k p_k)^T p_k| \leq -\eta g_k^T p_k, \tag{2.5b}$$

where $0 < \mu \leq \eta < 1$ and $\mu \leq \frac{1}{2}$. Condition (2.5*a*) ensures a "sufficient decrease" in F, and (2.5*b*) guarantees satisfaction of (2.4).

A popular alternative linesearch technique is known as *backtracking* (see, e.g., Dennis and Schnabel, 1983). Given a fixed $0 < \rho < 1$, a sequence $\{\beta_j\}$ is generated that satisfies $\beta_0 = 1$ and

$\beta_j > \beta_{j+1} \geq \rho\beta_j$. The value of α_k is taken as the first element in the sequence $\{\beta_j\}$ satisfying the sufficient decrease criterion

$$F(x_k) - F(x_k + \beta_j p_k) \geq -\mu\beta_j g_k^T p_k,$$

where $0 < \mu < 1$. Since a backtracking method can never generate a value of α greater than unity, *(2.4) may not hold.* Most implementations of unconstrained quasi-Newton methods with a backtracking linesearch simply skip the update in this case. (As the iterates converge, (2.4) is satisfied for the initial step of unity, and hence the difficulty does not arise in the limit.) The simplicity of backtracking algorithms and their utility in convergence proofs have led to their frequent appearance in the literature.

3. Methods for nonlinear equality constraints

3.1. Basic theory and notation. In this section, we consider methods for problems that contain only nonlinear *equality* constraints, i.e.

$$\text{NEP} \qquad \begin{aligned} &\underset{x\in\mathbb{R}^n}{\text{minimize}} && F(x) \\ &\text{subject to} && c_i(x) = 0, \quad i = 1, \ldots, m. \end{aligned}$$

We concentrate on this simplified problem in order to emphasize the treatment of nonlinear constraints.

The Kuhn-Tucker conditions for NEP state the existence of an m-vector λ^* (the *Lagrange multiplier vector*) such that

$$g(x^*) = A(x^*)^T\lambda^*. \tag{3.1}$$

(For a detailed discussion of first- and second-order Kuhn-Tucker conditions for optimality, see, for example, Fiacco and McCormick, 1968, and Powell, 1974.)

Let $Z(x)$ denote a matrix whose columns form a basis for the set of vectors orthogonal to the rows of $A(x)$; i.e., $A(x)Z(x) = 0$. An equivalent statement of (3.1) in terms of Z is

$$Z(x^*)^T g(x^*) = 0.$$

The vector $Z(x)^T g(x)$ is termed the *projected gradient* of F at x.

The *Lagrangian function*

$$L(x, \mu) = F(x) - \mu^T c(x),$$

where μ is an m-vector of Lagrange-multiplier estimates, plays an important role in understanding and solving constrained problems. Condition (3.1) is a statement that x^* is a *stationary point* (with respect to x) of the Lagrangian function when $\mu = \lambda^*$. One of the second-order sufficiency conditions for optimality is that the *projected Hessian of the Lagrangian function,*

$$Z(x)^T\nabla^2 L(x, \mu)Z(x) = Z(x)^T\Big(\nabla^2 F(x) - \sum_{i=1}^{m} \mu_i \nabla^2 c_i(x)\Big) Z(x),$$

is positive definite when $x = x^*$, $\mu = \lambda^*$.

In the following, we consider *sequential quadratic programming* (SQP) methods for NEP, in which the search direction is the solution of a *quadratic programming subproblem* and the steplength achieves a sufficient reduction in some "merit function". The purpose of the merit function is to enforce steady progress to the solution by balancing the (usually) conflicting aims of reducing the objective function and satisfying the nonlinear constraints.

The quadratic programming subproblem is of the form:

$$\underset{p \in \mathbb{R}^n}{\text{minimize}} \quad g_k^T p + \frac{1}{2} p^T H_k p \tag{3.2a}$$

$$\text{subject to} \quad A_k p = -c_k, \tag{3.2b}$$

where c_k and A_k denote c and A evaluated at x_k. The so-called *linearized* constraints (3.2*b*) represent a first-order approximation to the nonlinear constraints of the original problem. The matrix H_k is an approximation to the Hessian of the Lagrangian function. The Lagrange multiplier vector of this subproblem (denoted by μ_k) satisfies

$$H_k p_k + g_k = A_k^T \mu_k,$$

and may be used as an estimate of λ^*.

SQP methods differ in their definitions of the matrix H_k, and, as we shall see later, formulation of the QP constraints (3.2*b*). In the next two sections, we shall see how the choice of matrix H_k is related to the method used to solve the equality-constraint QP (3.2).

3.2. Methods for equality-constraint QP. All methods for solving (3.2) may be viewed as alternative methods for solution of the *augmented system* of equations for p and μ

$$\begin{pmatrix} H & A^T \\ A & \end{pmatrix} \begin{pmatrix} p \\ -\mu \end{pmatrix} = -\begin{pmatrix} g \\ c \end{pmatrix}, \tag{3.3}$$

which expresses the optimality and feasibility conditions. (The subscript k has been suppressed for convenience.)

Methods for solving (3.3) are often based upon constructing an equivalent, but simpler, system. Let S be a nonsingular $(n+m) \times (n+m)$ matrix. The solution of (3.3) is equivalent to the solution of

$$S^T \begin{pmatrix} H & A^T \\ A & \end{pmatrix} S \begin{pmatrix} \bar{p} \\ -\bar{\mu} \end{pmatrix} = -S^T \begin{pmatrix} g \\ c \end{pmatrix} \quad \text{with} \quad \begin{pmatrix} p \\ -\mu \end{pmatrix} = S \begin{pmatrix} \bar{p} \\ -\bar{\mu} \end{pmatrix}. \tag{3.4}$$

We shall consider two commonly used choices for S derived from the LQ factorization of A: an $m \times m$ lower-triangular matrix L and an $n \times n$ matrix Q such that

$$AQ = (L \quad 0). \tag{3.5}$$

Assume that the columns of Q are partitioned so that

$$Q = (Y \quad Z),$$

where Y has m columns. Then let S be given by

$$S = \begin{pmatrix} Y & Z & \\ & & I \end{pmatrix}, \tag{3.6}$$

and let p_Y and p_Z denote the first m and last $n - m$ elements of $\bar{p}$, respectively. Substituting from (3.6) into (3.4), we obtain

$$\begin{pmatrix} Y^THY & Y^THZ & L^T \\ Z^THY & Z^THZ & \\ L & & \end{pmatrix} \begin{pmatrix} p_Y \\ p_Z \\ -\mu \end{pmatrix} = - \begin{pmatrix} Y^Tg \\ Z^Tg \\ c \end{pmatrix}.$$

Thus, p and μ may be found by solving the equations

$$Lp_Y = -c \tag{3.7a}$$

$$Z^THZp_Z = -Z^Tg - Z^THYp_Y \tag{3.7b}$$

$$p = Yp_Y + Zp_Z \tag{3.7c}$$

$$L^T\mu = Y^T(g + Hp). \tag{3.7d}$$

Note that the projected Hessian matrix Z^THZ appears explicitly in (3.7*b*). If (3.2) has a well-defined solution, this matrix is positive definite.

We consider two definitions of L and Q. In the methods of Gill and Murray (1974), Wright (1976), Murray and Wright (1978) and Gill *et al.* (1984c), L and Q are found by explicitly triangularizing A using Householder matrices or stabilized elementary matrices. In this paper, we consider only the use of Householder matrices, in which case the matrix Q is orthogonal.

Computation of the LQ factorization may be viewed as *updating* an existing factorization as new rows are added in the last position. Assume that the LQ factorization (3.5) of A is available, and consider the matrix $\bar{A}$, which is A augmented by the row a^T. Then

$$\bar{A}Q = \begin{pmatrix} A \\ a^T \end{pmatrix} Q \equiv \tilde{L} = \begin{pmatrix} L & 0 \\ t^T & s^T \end{pmatrix}, \tag{3.8}$$

where t and s are the relevant partitions of Q^Ta. Let $\tilde{Q}$ denote a Householder matrix of the form

$$\tilde{Q} = I - \frac{1}{\beta}uu^T,$$

where the vector u and scalar β are chosen to annihilate all but the first element of s, and to leave t unchanged. (For details of how these quantities are defined, see Stewart, 1973.) Then

$$\tilde{L}\tilde{Q} = \begin{pmatrix} L & 0 & 0 \\ t^T & \tau & 0 \end{pmatrix} \tag{3.9}$$

$$\text{or} \qquad \bar{A}\bar{Q} \equiv (\bar{L} \quad 0), \quad \text{where} \quad \bar{Q} = Q\tilde{Q}.$$

The so-called "standard" LQ factorization is a version of (3.8) and (3.9) in which the rows of A are added to the null matrix one by one. The initial Q matrix is taken as the identity, and the initial L is the null matrix. While computing the factorization, the sequence of Householder transformations is stored *in compact form* (i.e., Q is not stored explicitly); the vector a^TQ needed in (3.8) is obtained by applying the sequence of stored transformations. Once the initial factorization has been completed, the necessary explicit matrix Q is obtained by multiplying the compact Householder transformations together in reverse order.

A second choice for the matrices L and Q involves defining Q so that

$$Q^THQ = I. \tag{3.10}$$

In this case, $Z^THZ = I$ and $Z^THY = 0$, and the equations (3.7) for p and μ become

$$Lp_Y = -c \tag{3.11a}$$

$$p_Z = -Z^Tg \tag{3.11b}$$

$$p = Yp_Y + Zp_Z \tag{3.11c}$$

$$L^T\mu = Y^Tg + p_Y. \tag{3.11d}$$

The recurrence relations (3.8) and (3.9) may be adapted to compute Q satisfying (3.10) by defining the initial Q matrix to be R^{-1}.

Equations (3.11) are used to solve the augmented system in the QP method of Goldfarb and Idnani (1983). Similar techniques have been suggested previously for both positive-definite and indefinite quadratic programs. In the latter case, the relationship $Z^THZ = D$ is maintained instead of (3.10), where the matrix D may be diagonal (Murray, 1971) or block-diagonal (Bunch and Kaufman, 1978).

3.3. Properties of the SQP search direction. It is clear from (3.7c) that the search direction is the sum of two vectors: a range-space component p_R ($\equiv Yp_Y$), and a null-space component p_N ($\equiv Zp_Z$). The range-space vector satisfies the underdetermined equations (3.2b), and thus defines a step to the linearized versions of the nonlinear constraints. (If the columns of Y are orthogonal, p_R defines the step to the *nearest* point on the linearized constraints.) The null-space component p_N defines the step from $x_k + p_R$ to the minimum of the quadratic model of the Lagrangian function in the subspace orthogonal to the constraint normals. An explicit distinction between the aims of satisfying the constraints and minimizing the Lagrangian function is important because the properties of the equations that define the associated vectors are essentially different. The range-space vector is a *Newton step* in the sense that it is computed using exact derivative information. By contrast, the null-space component is a *quasi-Newton step* defined using approximate derivative information from H_k. The better accuracy of the range-space component implies that $\|c\|$ generally remains smaller than $\|Z^Tg\|$ as the solution is approached. During the final iterations, the behavior of SQP methods is characterized by the relationship

$$\|p_Y\|/\|p_Z\| \rightarrow 0; \tag{3.12}$$

i.e., the final search directions lie almost wholly in the null space of A.

3.4. The definition of H_k. Based on its success in the unconstrained case, the BFGS formula (2.3) seems a logical choice for updating an approximation to the Hessian of the Lagrangian function. However, the definition of the updating formula in the constrained case is complicated by the fact that there is some choice as to *which* matrix should be approximated.

An important feature of the BFGS update in unconstrained optimization is the maintenance of positive-definiteness. In the constrained case, the relevant positive-definite matrix is the *projected* Hessian Z^THZ. Accordingly, the first class of methods that we shall consider is based on maintaining a quasi-Newton approximation H_Z to the projected Hessian. There are many closely-related variants of this approach. For example, for linearly constrained problems, see Gill and Murray (1974) and Murtagh and Saunders (1978). For nonlinearly constrained problems, see Murray and Wright (1978), Coleman and Conn (1982), Gabay (1979), and Nocedal and Overton (1982). A typical update for methods in this class is

$$\bar{H}_Z = H_Z - \frac{1}{u_Z^T s_Z} u_Z u_Z^T + \frac{1}{y_Z^T s_Z} y_Z y_Z^T, \tag{3.13}$$

where barred quantities refer to the updated values, $g_Z = Z^T g$, $y_Z = \bar{g}_Z - g_Z$, $s_Z = Z^T(\bar{x} - x)$ and $u_Z = H_Z s_Z$. For these projected quasi-Newton methods, the matrix H that defines the QP subproblem (3.2) is ZH_ZZ^T, which is positive semi-definite. A common feature of the projected quasi-Newton methods mentioned above is that the null-space component of the search direction is defined from the equations

$$H_Z p_Z = -g_Z$$

in order to avoid the necessity of recurring the matrix Z^THY (*cf.* (3.7*b*)).

In contrast to the unconstrained case, it is not always possible to choose a step length that guarantees the condition $y_Z^T s_Z > 0$. As a result, there may be iterations in which the quasi-Newton update cannot be performed because of loss of positive-definiteness. However, since (3.12) implies that the search directions will lie in the null space of A as the iterates converge, $y_Z^T s_Z$ is typically positive in the neighborhood of the solution.

An immediate consequence of storing only an approximation to the *projected* Hessian is that the QP multipliers μ cannot be computed, since the full matrix H is not available (*cf.* (3.7*d*)). However, if Q is orthogonal, the least-squares multipliers λ_L at a point x may be calculated from the equations

$$L^T\lambda_L = Y^T g(x).$$

The continuity properties of the associated Z are significant in projected quasi-Newton methods because Z defines the operation of projection. For example, in proving local convergence for algorithms that explicitly utilize Z, it is essential that small changes in x should lead to small changes in Z (see, e.g., Coleman and Conn, 1982a, b; and Nocedal and Overton, 1982). The

standard method of computing the LQ factorization does not provide a continuous representation of $Z(x)$ (see Coleman and Sorensen, 1984). However, a continuous representation of Z may be defined using a trivial modification of the relations (3.8) and (3.9). Recall that, given the factorization (3.5) of the matrix A, we require the factors $\bar{L}$ and $\bar{Q}$ of $\bar{A}$ at the point $\bar{x} = x + \alpha p$. Exactly as in the standard factorization, (3.8) and (3.9) can be used to update the factors as the rows of $\bar{A}$ are added to the null matrix one by one, except that the explicit matrix Q *from the previous QP subproblem* is taken as the initial matrix Q in (3.8). Each Householder transformation is then multiplied into Q after the corresponding row has been transformed. With the standard procedure, the Householder transformations can be stored in compact form, but with this approach, each new row of $\bar{A}$ must be transformed by a full orthogonal matrix (which is somewhat more expensive unless some of the constraints are linear). The benefit is that Z is continuous at a point where A has full rank, and approaches a limit when computed at a sequence of points $\{x_k\}$ converging sufficiently fast to a suitable point x^* (see Gill *et al.*, 1985a).

The second major representation of H_k is as a quasi-Newton approximation to the *full* Hessian of the Lagrangian function. (This method is especially appropriate if the calculation of the search direction requires the full Hessian, e.g., the method defined by equations (3.11).) Consider a BFGS-type update of the form

$$\bar{H} = H - \frac{1}{u^T s} uu^T + \frac{1}{v^T s} vv^T, \tag{3.14}$$

where $s = \bar{x} - x$ and $u = Hs$. Since H is meant to approximate the Hessian of the Lagrangian function, a "natural" choice for v would be y_L, the difference in gradients of the Lagrangian function; i.e.,

$$y_L = \bar{g} - g - (\bar{A}^T - A^T)\lambda,$$

with λ taken as the best available multiplier estimate. However, since x^* is not an unconstrained minimum of the Lagrangian function, it may be *impossible*, with *any* linesearch, to find a step length for which $y_L^T s$ is positive. Hence, the update might be skipped at every iteration, which would not only destroy the local convergence properties, but also adversely affect the efficiency of the method away from the solution.

A popular method for dealing with this difficulty is to use y_L as v in (3.14) only when $y_L^T s$ is sufficiently positive; otherwise, v is taken as a perturbed vector $\bar{y}_L$ such that $\bar{y}_L^T s > 0$. A perturbation that we have found to be quite successful in practice is defined as follows. When $y_L^T s < 0$, compute the scalar

$$\bar{\omega} = \frac{-y_L^T s}{s^T(\bar{A}^T\bar{c} - A^T c)}.$$

The quantity in the denominator is an approximation to the curvature of $\|c\|_2^2$, which is positive at x^*. If $\bar{\omega}$ is negative, the update is skipped; otherwise,

$$v = y_L + \omega(\bar{A}^T\bar{c} - A^T c),$$

where ω is any positive scalar such that $\omega > \bar{\omega}$. The motivation for this choice of v is the result (due to Boggs, Tolle and Wang, 1982) that a necessary condition for q-superlinear convergence is that the approximate Hessian matrices must satisfy

$$\lim_{k\to\infty} \frac{\|Z_k Z_k^T(H_k - \nabla^2 L(x^*, \lambda^*))p_k\|}{\|p_k\|} = 0.$$

3.5. Inconsistent and ill-conditioned constraints. In the preceding discussion of SQP methods, we have assumed that the equations (3.2*b*) are *consistent*. When all the constraints of the original problem are linear, the subproblem constraints can be inconsistent only if the original problem has no solution. With nonlinear constraints, however, *the constraints of the subproblem may be inconsistent even when the original problem has a well-posed solution*. Techniques for dealing with inconsistent constraints fall into two major categories. In both cases, the search direction satisfies a *shifted* set of linear constraints

$$Ap = r - c. \tag{3.15}$$

The first approach generates a search direction designed to minimize a weighted combination of the quadratic approximation to the Lagrangian and the residual vector of the unperturbed constraints (the vector $c + Ap$). The compromise is achieved by choosing p as the solution of an unconstrained problem of the form

$$\underset{p}{\text{minimize}}\ \sigma\left(g^T p + \frac{1}{2}p^T H p\right) + \Phi(c + Ap), \tag{3.16}$$

where $\Phi(r)$ is a scalar-valued function that measures the "size" of r, and σ is a non-negative scalar. The theoretical basis of this class of method is that a minimum of (3.16) always exists, even when a solution of the QP subproblem (3.2) does not.

The algorithms of Biggs (1972a), Bartholomew-Biggs (1982) and Fletcher (1981) correspond to choosing Φ as the two-norm and one-norm, respectively. In either case, it can be shown that the unconstrained minimizer of (3.16) is the solution of a QP with objective function (3.2*a*) and constraints (3.15). The form of the vector r depends on the definition of Φ. If $\Phi(r)$ is defined as $\frac{1}{2}\|r\|_2^2$, all the components of r are generally non-zero, and p is defined by equations similar to (3.11) (see Bartholomew-Biggs, 1982, for more details). If $\Phi(r) = \|r\|_1$, p is the solution of an ℓ_1-QP (see Fletcher, 1981). At the solution of this problem, a subset of the original constraints (3.2*b*) will be satisfied exactly (i.e., the corresponding components of r will be zero). This approach therefore implicitly "discards" some of the violated constraints from the subproblem.

The methods of Bartholomew-Biggs and Fletcher are based on the properties of penalty functions (see Fletcher, 1983, for a survey). A feature of these treatments of inconsistent constraints is that σ is always non-zero in (3.16). An alternative approach is to define the composite function only when the constraints are found to be inconsistent. Other SQP methods with a strategy of

this type have been suggested by Powell (1977), Schittkowski (1983), Tone (1983) and Gill *et al.* (1984b).

The second technique for the treatment of inconsistency is based on the observation that only the range-space portion of the QP search direction is ill-defined when the constraints are incompatible. With this approach, a well-defined procedure is used to compute p_R, and p_Z and p are obtained from (3.7*b*) and (3.7*c*). The most straightforward application of this approach is to define p_R as a solution of the linear least-squares problem

$$\underset{p_R}{\text{minimize}} \quad \|c + Ap_R\|_2, \tag{3.17}$$

which gives r as the *smallest* constraint shift (in the least-squares sense). This choice of p_R is equivalent to computing the first iterate of the Gauss-Newton method for minimizing the two-norm of the *nonlinear* constraint violations. Therefore, the strategy for dealing with the constraints has changed from the possibly unsolvable problem of finding a point such that $c(x) = 0$ to the always solvable problem of minimizing $\sum c_i^2(x)$.

Since A must be rank-deficient when the constraints are inconsistent, the solution of (3.17) is not unique. A suitable choice of p_R in this case is the minimum-length solution, which can be computed using the *complete orthogonal factorization:*

$$PAQ = \begin{pmatrix} L & 0 \\ 0 & 0 \end{pmatrix},$$

where P and Q are orthogonal matrices and L is a lower-triangular matrix whose dimension is equal to the rank of A.

Unfortunately, neither of these techniques resolves the difficulties caused by constraints that are *almost* inconsistent (i.e., ill-conditioned). Ill-conditioning in A will tend to cause p_R to be large in norm (see (3.7*a*)). In these situations it is necessary in practice to limit the norm of p. It might appear that the least-length solution of (3.17) would automatically be satisfactory. However, the computation of p using the complete orthogonal factorization involves serious practical difficulties — in particular, a strategy must be included for estimating the rank of A. It is well known that the definition of numerical "rank" is problem-dependent. The rank can never be determined without making an explicit judgment about scaling, i.e., a decision as to which quantities can be considered "negligible". The choice of rank is critical in the Gauss-Newton method because a slight alteration in the value of the tolerance used to estimate the rank may lead to completely different behavior.

If the composite function (3.16) is used, an *explicit* bound on the norm of p may be enforced by temporarily imposing additional constraints on the problem. (This type of procedure is used within trust-region algorithms for unconstrained optimization.) The effect of the trust-region constraints is to modify (implicitly or explicitly) the derivative information that defines the search direction. For example, if a temporary bound is placed on the two-norm of p, the search direction

satisfies equations in which the second-derivative approximation is modified by a multiple of the identity matrix. Thus, Z^THZ and A are implicitly modified — an unfortunate result, since we would prefer the projected Hessian approximation to be independent of the conditioning of A.

The development of stable robust methods for dealing with ill-conditioned constraints is still an active area of research. One possible approach was suggested in Section 1.2, where small perturbations of constraints were used to resolve inconsistencies caused by modelling inaccuracies. By changing the constraints (3.2*b*) to suitable perturbed inequality constraints, (3.2) always has a bounded solution. For example, consider defining p_R as the solution of the *inequality-constrained* quadratic program

$$\begin{aligned} &\underset{p_R \in \mathbb{R}^n}{\text{minimize}} && \tfrac{1}{2} p_R^T p_R \\ &\text{subject to} && -\delta \leq Ap_R + c \leq \delta, \end{aligned} \tag{3.18}$$

where δ is a vector of small quantities that are forced to approach the feasibility tolerances for the original constraints as x approaches x^*. The subset of constraints active at the solution of (3.18) may then be used to define Z and Y, from which p_z and p can be computed using (3.7).

4. Methods for nonlinear inequality constraints

4.1. Background. In the final problem to be considered, all the constraints are nonlinear *inequalities*:

NIP
$$\begin{aligned} &\underset{x \in \mathbb{R}^n}{\text{minimize}} && F(x) \\ &\text{subject to} && c_i(x) \geq 0, \quad i = 1, \ldots, m. \end{aligned}$$

We consider this simplified form in order to concentrate on the treatment of nonlinear inequality constraints.

Let the matrix $A(x)$ denote the Jacobian of $c(x)$. The constraint c_i is said to be *active* at x if $c_i(x) = 0$, and *violated* if $c_i(x) < 0$. The Kuhn-Tucker conditions for NIP are similar to those for the equality-constraint case, except that they involve only constraints active at x^*, and impose a sign restriction on the Lagrange multipliers. The major difference between inequality- and equality-constrained problems is that the set of constraints active at the solution is *unknown* in the inequality case. Therefore, algorithms for NIP must include a procedure (termed an *active-set strategy*) that determines the correct active set — usually, by maintaining a *working set* that estimates the active set. In this section we discuss the additional algorithmic complexity in SQP methods that arises specifically from the presence of inequality constraints.

4.2. Formulation of the QP subproblem. Broadly speaking, two extreme types of QP subproblems can be posed when solving inequality-constrained problems. The first — called an *IQP strategy* — corresponds to representing *all* nonlinear inequality constraints as *inequalities*

in the QP subproblem; this has been by far the most widely used formulation in published SQP methods. The *standard IQP search direction* is the solution of

$$\underset{p\in\Re^n}{\text{minimize}} \quad g^Tp + \tfrac{1}{2}p^THp \tag{4.1a}$$

$$\text{subject to} \quad Ap \geq -c, \tag{4.1b}$$

where H is an approximation to the Hessian of the Lagrangian function and A is the Jacobian of $c(x)$ evaluated at the current iterate. In general, the solution of (4.1) must be found by iteration. Thus, the structure of an SQP method with an IQP strategy involves *major* and *minor* iterations -- the minor iterations being those of the quadratic programming method.

Because (4.1) includes all the constraints of NIP, it is convenient to take *the active set of the QP* as a prediction of the active set of the nonlinearly constrained problem. The theoretical justification for this strategy is that the QP (4.1) will make a correct prediction of the active set of the original problem in a neighborhood of x^* for *any* bounded positive-definite matrix H (Robinson, 1974). Furthermore, the multipliers of (4.1) approach the multipliers of NIP as the iterates converge to x^*, and hence it is common to take the QP multipliers as the next multiplier estimate.

The second extreme form of subproblem in SQP methods involves a QP with only *equality* constraints. In order to use an *EQP strategy*, some determination must be made before posing the QP as to which constraints are to be included. An EQP method should have the property that it will select the correct active set in some neighborhood of x^*. Therefore, such methods tend to choose constraints that satisfy properties of the active constraints in a neighborhood of the solution -- e.g., are "small" in magnitude, or satisfy the sign requirements for the Lagrange multiplier estimates. A benefit of an EQP method is that, in general, the subproblem will be *easier to solve* than one with inequality constraints.

To a large extent, the active-set strategy will determine the choice of quasi-Newton update and Lagrange multiplier estimate. For example, if an IQP strategy is used, the method used to solve (4.1) will require specification of the full matrix H. On the other hand, an EQP strategy is usually implemented with an approximation to the projected Hessian. The following table summarizes the major features of the two active-set strategies.

EQP	**IQP**
• QP subproblem: $\underset{p\in\Re^n}{\text{minimize}} \; g^Tp + \frac{1}{2}p^THp$ subject to $\hat{A}p = -\hat{c}$.	• QP subproblem: $\underset{p\in\Re^n}{\text{minimize}} \; g^Tp + \frac{1}{2}p^THp$ subject to $Ap \geq -c$.
• Least-squares multipliers.	• QP multipliers.
• Projected Hessian approximation.	• Full Hessian approximation.

These two active-set strategies are the extremes of a whole range of possibilities. Other methods have been defined that use features from both approaches. For example, the method of

Schittkowski (1981) solves a QP subproblem of the form (4.1), but uses a pre-assigned working set to specify which of the constraint gradients should be recomputed for the next major iteration. Similarly, it is possible to treat linear constraints with an EQP approach and nonlinear constraints with an IQP approach (see Gill *et al.*, 1984d).

It is important to note that IQP methods can be implemented so that, as the solution is approached, the amount of overhead per major iteration is the same as for an EQP method (i.e., solution of a single set of equations of the form (3.3)). This can be achieved by solving (4.1) with a QP method that allows the active set from one subproblem to be used to initialize the next. Since the active set of the subproblem eventually becomes the correct active set for the nonlinear problem, *QP subproblems near the solution reach optimality in only one minor iteration.*

4.3. Active-set strategies in quadratic programming methods. Quadratic programming methods for problems with inequality constraints solve a sequence of problems (3.3) in which the constraints in the working set are treated as *equalities*. The major differences among QP methods arise from the numerical procedures for solving the associated linear equations, and the strategies that control changes in the working set. (Modern QP methods are surveyed by Fletcher (1981) and Gill, Murray and Wright (1981).)

We briefly outline two methods of selecting constraints for the working set when solving (4.1). At each iteration, let p and λ denote the current estimates of the solution and optimal multiplier vector, and let r denote the residual vector $r \equiv Ap + c$. The "hat" notation indicates quantities associated with the working set. The vector δp is defined such that $p + \delta p$ is the solution of a QP with the original objective function, subject to the constraints of the working set held at equality, and $\delta\lambda$ denotes the corresponding change in the multiplier estimate. As indicated in Section 3.2, δp and $\delta\hat{\lambda}$ are the solution of the linear system

$$\begin{pmatrix} H & \hat{A}^T \\ \hat{A} & 0 \end{pmatrix} \begin{pmatrix} \delta p \\ -\delta\hat{\lambda} \end{pmatrix} = - \begin{pmatrix} g_Q - \hat{A}^T\hat{\lambda} \\ \hat{r} \end{pmatrix}, \tag{4.2}$$

where g_Q denotes $g + Hp$, the gradient of the quadratic function. The algorithms considered here always attempt to move from the minimum on one working set to the minimum on another by taking steps of the form $p + \delta p$ and $\lambda + \delta\lambda$. However, the maintenance of certain properties of the working set can cause a step α $(0 \leq \alpha < 1)$ to be taken, where α depends upon the active-set strategy being used.

In an *active-set feasible-point QP method*, p is feasible ($r_i \geq 0$ for all i), but λ is not dual-feasible (i.e., $\lambda_i < 0$ for at least one i). Changes in the working set are designed to maintain feasibility of p, but to move interior to constraints that have negative Lagrange multipliers. At a typical iteration, the working set comprises the constraints satisfied exactly at p (i.e., $\hat{r} = 0$ in (4.2)). If $p + \delta p$ remains feasible (i.e., $r_i + a_i^T\delta p \geq 0$ for i not in the working set), then the full step of unity is permitted. A constraint with a negative multiplier (usually, the *most* negative) is then deleted from the working set. Otherwise, α is taken as the smallest step such that the residual of a constraint not in the working set becomes zero at α, and the corresponding constraint is *added*

to the working set. For more details concerning the implementation of feasible-point quadratic programming methods, see Gill and Murray (1978) and Gill *et al.* (1984a, 1985b).

The second strategy is typical of *dual-feasible active-set methods*. In these methods, p is not feasible (i.e., some $r_i < 0$) but λ is always dual-feasible (all $\lambda_i \geq 0$). Changes in the working set are designed to maintain non-negative multipliers while moving to satisfy the violated constraints. At the beginning of a typical iteration, all the constraints in $\hat{A}$ are satisfied exactly except one (i.e., $\hat{r}$ is a multiple of a unit vector in (4.2)). The step length α is taken as one if $p + \delta p$ is dual-feasible (i.e., $\lambda_i + \delta\lambda_i \geq 0$). Otherwise, α is chosen as the largest step such that $\lambda_i + \alpha\delta\lambda_i = 0$ for an index i in the working set, and the corresponding constraint is deleted from the working set. After a unit step is taken, a constraint with a negative residual (usually, the most negative) is added to the working set. (Note that we have given a considerably simplified description of the dual-feasible iteration in order to emphasize the similarities between dual- and primal-feasible methods. In practice, δp and $\delta\lambda$ are not found directly from (4.2) because the new constraint may be dependent on the constraints already in the working set.) For further information concerning the implementation of dual-feasible quadratic programming methods, see Goldfarb and Idnani (1983) and Powell (1983b).

For both of these active-set strategies, each change in the working set leads to a simple change to $\hat{A}$, which in turn leads to a change in the factorizations used to solve (4.2).

Both of the active-set strategies described require an *initialization procedure* to obtain an initial primal- or dual-feasible point. As noted above, for efficiency within an SQP method, it is critical that this procedure should be able to utilize a pre-assigned working set.

The initialization procedure for the primal feasible-point method is equivalent to a linear programming problem. Consider the sum of infeasibilities

$$v(p) = - \sum_{a_i^T p + c_i < 0} (a_i^T p + c_i).$$

Note that $v(p)$ is a linear function that is zero at any feasible point, and positive at an infeasible point. Therefore, a feasible point can be found by minimizing $v(p)$, subject to continuing to satisfy the constraints with positive residuals at p. The function $v(p)$ may be minimized using an active-set strategy that is almost identical to that of the feasible-point active-set method. The principal differences are that the search direction is defined as $-ZZ^T\nabla v(p)$, and α is chosen as $\min(\alpha_1, \alpha_2)$, where α_1 is the maximum step that can be taken without violating one of the constraints that is currently satisfied, and α_2 reaches the furthest constraint along δp that is currently violated. (Several violated constraints may become satisfied during a single iteration.) For efficiency, the implementation of this procedure should reflect the similarity of the linear algebraic computations associated with iterations in both the feasibility and QP phases — in particular, each iteration involves an update of the same factorization of the working set. The computations in both phases may be performed by exactly the same program modules. The two-phase nature of the

algorithm is reflected by changing the function being minimized from the sum of infeasibilities to the quadratic objective function. An important feature of this type of implementation is that if the pre-assigned working set is similar to the active set, just a few changes in working set are necessary to achieve feasibility. In particular, if the initial point is feasible, the procedure merely computes all the relevant factorizations (which are also needed for the QP iterations) and performs a feasibility check.

If a dual-feasible active-set strategy is used, the following initialization procedure may be employed. The procedure is based on finding a subset of the pre-assigned working set on which the multipliers are positive. First, the minimum of the quadratic on the pre-assigned working set is computed by solving (4.2) with $p = 0$ and $\lambda = 0$. If the $\delta\lambda_i$ are non-negative, the initial point can be taken as $p = \delta p$ and $\lambda = \delta\lambda$. Otherwise, a constraint with a negative multiplier is deleted, the factorizations are updated and (4.2) is solved again. This process is repeated until all the multipliers are non-negative or the working set is empty, in which case $p = \delta p$ and $\lambda = \delta\lambda$ define the required initial point. (Note that the unconstrained minimum is trivially dual feasible.)

An alternative initialization procedure is to start at the unconstrained minimum and give preference to adding the pre-assigned constraints. However, if the pre-assigned working set is similar to the active set, this scheme is likely to require more work than the procedure above. First, more operations are required to compute the factorizations by updating. Second, even if the pre-assigned working set defines the optimal feasible point, the number of QP iterations may not be equal to the dimension of the optimal working set, since it cannot be guaranteed that the multipliers will remain dual-feasible during the intermediate iterations.

4.4. Conditioning of the working set. One of the most important issues in the implementation of QP algorithms is *robustness*. During the solution of a nonlinear problem, quadratic subproblems of wildly varying degrees of difficulty are generated "automatically". Even if the original nonlinear problem is well-conditioned in the neighborhood of the solution, the QP subproblems of the early iterations may be very badly behaved. The most common difficulties include singular or nearly rank-deficient Jacobians, subproblems with very small feasible regions and severely ill-conditioned Hessian matrices. (For examples, see Section 5.)

One of the most critical features of a QP implementation is the strategy for maintaining a well-conditioned working set. The spectral condition number of A provides a measure of the degree of independence of the constraints in the working set. This number (the ratio of the largest to smallest singular values of A) will decrease when a constraint is deleted from the working set and increase when a constraint is added. The worst case occurs when the new working set is singular, i.e., an attempt is made to add a constraint that is dependent on constraints already in the working set. However, if a near-dependent constraint is added to the working set, the condition number may increase substantially. Accordingly, it is important that the constraint-selection procedure should consider the condition number of the new working set.

With exact arithmetic and a non-singular initial working set, the active-set strategy de-

scribed above for the primal-feasible method would never generate a singular working set. To see why, recall that $A\delta p = 0$ at every iteration. Thus, δp will never intersect a constraint that is exactly linearly dependent upon A. In practice, of course, the difficulty arises when the candidate constraints are *nearly* dependent. Determination of the condition number requires the singular values of A, which would be too expensive to compute. Instead, a QP method can use an inexpensive *condition estimator* — for example, the ratio of the largest to smallest diagonals of the LQ factor L (see (3.5)), which is a *lower bound* on the condition number of A.

Exercising control over this condition estimator turns out to particularly easy in a primal-feasible active-set method, if it is acceptable to violate constraints by a small tolerance. Suppose that each constraint has an associated user-defined tolerance that specifies the maximum permissible constraint violation. Let α_m denote the maximum step at which $p + \alpha_m \delta p$ does not violate any constraint by more than its feasibility tolerance. All constraints at distance α ($\alpha \leq \alpha_m$) from the current point are then viewed as acceptable candidates for inclusion in the working set. A criterion that we have found to be particularly successful in practice (due to Harris, 1973) is to add the constraint whose normal makes the largest angle with the search direction. In the case where the null space of A is of dimension one (for example, in the simplex method for linear programming), this choice gives the smallest condition estimator over the candidate set.

An unsatisfactory feature of the Harris scheme is that all the constraints active at the solution tend to be *violated* by their feasibility tolerances, even when the final active set is not ill-conditioned. However, this idea can be generalized so that constraint violations by δ are permitted when necessary to improve the conditioning of the working set, but an attempt is also made to minimize the constraint violations. With this strategy, the constraints active at the solution tend to be satisfied *exactly* rather than violated (see Gill *et al.*, 1985b). (An interesting result is that *negative* steps are sometimes necessary.)

5. Sample runs

In this section we shall use several examples to illustrate the performance of nonlinear programming software on practical problems. All the problems were solved in double precision on an IBM 3081 using the VS Fortran compiler with optimization level 3.

5.1. A comparison of IQP and EQP methods. The purpose of the first set of runs is to illustrate the properties of methods based on the EQP and IQP active-set strategies. Two specific methods for *linearly* constrained optimization are considered. In order to aid the comparison, the methods have several features in common. Both methods recur an orthogonal factorization of the constraints in the working set and begin by computing a feasible point. Both methods treat simple bounds and linear constraints separately.

The first method is a standard IQP method for linearly constrained optimization. The Hessian of each QP subproblem is a positive-definite BFGS approximation to the full Hessian of the objective function. At each iteration, a steplength is computed that satisfies the linesearch

conditions (2.5*a*) and (2.5*b*). Each QP subproblem is solved using a feasible-point active-set method with an orthogonal factorization of the constraints in the working set. The final working set from the QP of one iteration is used as the initial working set for the next. For additional details concerning the Fortran implementation, the reader is referred to Gill *et al.* (1984b).

In our discussion, we shall refer to the output printed during the run. A single line of output is printed at the end of each major iteration. The major iteration number is given in the first column (marked "ITN"). The next column "ITQP" gives the number of minor iterations needed to solve the QP subproblem. The "STEP" column gives the step α_k taken along the computed search direction. "NUMF" is the total number of evaluations of the problem functions. "OBJECTIVE" is the value of the objective function, $F(x_k)$. Columns "BND" and "LC" give the numbers of simple-bound constraints and general linear constraints in the working set. "NZ" is the dimension of the null space of the current matrix of constraints in the working set. The next five entries give information about the derivatives of the problem at the current point. "NORM GF" is the two-norm of the free components of the objective gradient g_k, and "NORM GZ" is the two-norm of $Z_k^T g_k$. "COND H", "COND HZ" and "COND T" are estimates of the condition numbers of the Hessian, projected Hessian and matrix of constraints in the working set. "CONV" is a set of four logical variables C_1, C_2, C_3 and C_4, used to inform the user of the quality of the current estimate of the solution, with the following meanings. C_1 is true if the projected-gradient norm is small; C_2 is true if constraints are satisfied to within the user-specified tolerance; C_3 is true if the signs of the multipliers indicate optimality; and C_4 is true if the last change in x was small. Finally, in some of the runs an "S" is printed as the last item of the iteration summary. This indicates that it was necessary to skip the BFGS update to the approximate Hessian.

The second implementation is based on an EQP active-set strategy used in conjunction with a BFGS approximation of the projected Hessian $Z^T H Z$. The EQP is solved using the orthogonal LQ factorization and the Cholesky factorization of the projected Hessian (see (3.3) and (3.5)). If the objective function is decreasing at the step to the nearest satisfied constraint, the constraint is added to the working set. Lagrange multipliers are computed when the projected-gradient norm is less than some loose tolerance. If the smallest multiplier is negative, the corresponding constraint is deleted from the working set.

The iteration summary is the same as that for the IQP method except that only the condition estimate of the *projected* Hessian is printed (the full Hessian is not recurred), and additional information is printed about each change to the working set. When the status of a constraint changes, the index of the constraint is printed, along with the designation "L" (lower bound), "U" (upper bound) or "E" (equality). Indices 1 through "N" refer to the bounds on the variables, and the remaining indices refer to the general constraints. "KDEL" and "KADD" denote the indices of the constraints leaving and entering the working set. If an entry in one of these columns is zero, a constraint was not deleted or added. "MIN LM" is the multiplier associated with the constraint just deleted. If no constraint was deleted during the relevant iteration, the entry in this column is "--". The information printed in the "CONV" column is different from that given in the IQP

method. C_1 is true if the projected-gradient norm is smaller than some loose tolerance; C_2 is true if the projected-gradient norm is smaller than some tighter tolerance; C_3 is true if the change in the objective function was sufficiently small; and C_4 is true if the change in x is small. Note that the loose tolerance on the projected-gradient norm must be satisfied before any multipliers are computed and any constraint is deleted.

Figure 1 gives the output from runs on a well behaved seven-variable linearly constrained problem with seven general linear constraints and upper and lower bounds on the variables. At the solution, two bounds and three general linear constraints are active. A comparison of the output from the two runs will illustrate one of the major differences between IQP and EQP active-set strategies -- the different pattern of changes to the working set. In an EQP method, the problem functions are computed after every step along a search direction. Moreover, a constraint will be deleted from the working set only when the current point is considered to be sufficiently close to the minimum on the current working set (albeit to a very low accuracy). In other words, a constraint will be deleted only when the method has accumulated sufficient information about the *curvature* of the problem on the current working set. By constrast, many constraints may be added or deleted during a single major iteration of an IQP method. In the run given in Figure 1a, the IQP strategy finds a very close approximation to the correct active set during the *first* subproblem.

```
 ITN ITQP      STEP NUMF    OBJECTIVE BND  LC  NZ  NORM GF    NORM GZ COND NZ   COND H   COND T CONV
   0    5 0.0D-01    1 1.2127D 06   3   3   1 1.7D 03 8.2D 02 1.D 00 1.D 00 1.D 01 FFTT
   1    1 5.2D-01    3 9.5919D 05   3   3   1 1.1D 03 1.1D 02 1.D 00 2.D 00 1.D 01 FFTF
   2    2 1.0D 00    4 9.4923D 05   2   3   2 1.1D 03 2.5D 02 1.D 00 3.D 00 1.D 01 FTFF
   3    1 1.0D 00    5 9.3120D 05   2   3   2 1.1D 03 7.1D 01 1.D 00 2.D 00 1.D 01 FTTF
   4    1 1.0D 00    6 9.2960D 05   2   3   2 1.1D 03 1.5D 00 1.D 00 3.D 00 1.D 01 TTTF
   5    1 1.0D 00    7 9.2960D 05   2   3   2 1.1D 03 5.1D-02 1.D 00 3.D 00 1.D 01 TTTF
   6    1 1.0D 00    8 9.2960D 05   2   3   2 1.1D 03 1.2D-04 1.D 00 3.D 00 1.D 01 TTTF
   7    1 1.0D 00    9 9.2960D 05   2   3   2 1.1D 03 5.9D-07 1.D 00 3.D 00 1.D 01 TTTT

EXIT NP PHASE.  INFORM =  0   MAJITS =  7  NFEVAL =  9
```

Figure 1a. Results of an IQP method on a seven-variable linearly constrained problem.

```
 ITN JDEL  JADD      STEP NUMF    OBJECTIVE BND  LC  NZ  NORM GZ    MIN LM COND NZ   COND T   CONV
   0    0     0 0.0D-01    1 1.2127D 06   2   3   2 2.5D 02       --  1.D 00 7.D 01  T FFF
   1   5L   14L 1.0D-02    2 1.1945D 06   1   4   2 9.7D 02 -1.5D 03 1.D 00 7.D 01  F FTFS
   2    0    3L 1.6D-01    3 1.0755D 06   2   4   1 3.2D 02       --  1.D 00 7.D 01  T FTF
   3  11U    0  4.9D-01    5 9.5223D 05   2   3   2 1.7D 02 -2.6D 03 1.D 00 1.D 01  T FTF
   4    0    0  1.0D 00    6 9.4357D 05   2   3   2 2.2D 01       --  1.D 00 1.D 01  T FTF
   5   3L    0  1.0D 00    7 9.3434D 05   1   3   3 1.6D 02 -1.4D 02 8.D 00 2.D 01  T FTF
   6    0   4L  1.6D-01    8 9.2997D 05   2   3   2 3.7D 01       --  1.D 00 1.D 01  F FTFS
   7    0    0  1.0D 00    9 9.2960D 05   2   3   2 3.9D 00       --  1.D 00 1.D 01  F FTF
   8    0    0  1.0D 00   10 9.2960D 05   2   3   2 1.3D-03       --  1.D 00 1.D 01  T TTF
   9    0    0  1.0D 00   11 9.2960D 05   2   3   2 2.1D-06       --  1.D 00 1.D 01  T TTT

EXIT LC PHASE.  INFORM =  0   ITER =  9  NFEVAL =  11
```

Figure 1b. Results of an EQP method on a seven-variable linearly constrained problem.

During the final few iterations, the methods are essentially identical, with a single minor iteration being performed in the IQP method. Note that the asymptotic superlinear convergence rate is evident from the unit steplengths in the "STEP" column and the sequence of converging "NORM GZ" entries. This column may be used to verify the convergence to a local minimum (see Gill, Murray and Wright, 1981).

5.2. Are IQP methods superior to EQP methods? It is a popular myth that the more "opportunistic" IQP active-set strategy will tend to find the correct active set faster than an EQP strategy. The next two problems are intended to demonstrate that this is not always the case. On some problems the IQP method will be faster, on others it will be slower.

In Figures 2a and 2b we give the IQP and EQP results for the minimization of the six-variable nonlinear test function "Exp 6" subject to simple-bound constraints upon the variables (see Biggs, 1972b). The constraints are the vector of simple lower bounds $(0.5, 9, 0.9, 4, 3, 2)$ and simple upper bounds $(\infty, \infty, \infty, \infty, 4.9, \infty)$. The solution lies at the point $(1, 10, 1, 5, 4, 3)$, where no simple bounds are active. (For brevity, the output from some of the less important iterations has been omitted.)

Exp 6 is a problem for which the IQP method is substantially faster than the EQP method. Since no bounds are active at the solution, the ability of the quadratic subproblem to drop many of them during one major iteration allows the IQP method to identify the correct working set rapidly. By the seventh iteration, all the bounds have been deleted from the working set. An adequate solution is identified at iteration 32. By contrast, the requirement of some significant reduction in the projected gradient on each working set considerably slows down the rate at which the EQP method can delete constraints. It is not until iteration 55 that the correct working set is identified.

Cases where the more conservative strategy of the EQP method gives better performance are illustrated by the problem "Weapon" (see Bracken and McCormick, 1968). The objective function is

$$F(x) = \sum_{j=1}^{20} u_j \Big(\prod_{i=1}^{5} a_{ij}^{x_{ij}} - 1 \Big),$$

which is minimized subject to twelve general linear constraints and bounds on all the variables. At the solution, 75 bounds and seven linear constraints are active.

When applied to this problem, both methods require approximately the same number of function evaluations (see Figures 3a and 3b). However, the EQP method needs significantly less CPU time to obtain the solution -- 3.42 seconds compared to 19.48 seconds for the IQP method. The reason for this discrepancy is that the EQP method maintains a much better approximation to the working set. While moving from the initial feasible point (a vertex) to the solution, the EQP method maintains a projected Hessian approximation that never becomes larger than 19 (the size of the projected Hessian at the solution is 18). This performance is to be contrasted with that of the IQP method, for which the dimension of $Z^T H Z$ increased to 81 during the first

ITN	ITQP	STEP	NUMF	OBJECTIVE	BND	LC	NZ	NORM GF	NORM GZ	COND HZ	COND H	COND T	CONV
0	3	0.0D-01	1	1.8581D-01	2	0	4	1.1D 00	1.13D 00	1.D 00	1.D 00	1.D 00	FTTT
1	2	1.1D-01	3	1.1602D-01	1	0	5	6.4D-01	6.39D-01	9.D 00	4.D 00	1.D 00	FTTF
2	1	1.0D 00	4	6.1503D-02	1	0	5	3.2D-01	3.21D-01	8.D 00	4.D 00	1.D 00	FTFF
3	1	1.0D 00	5	1.6543D-02	1	0	5	1.2D-01	1.17D-01	1.D 01	4.D 00	1.D 00	FTTF
4	1	1.0D 00	6	1.2345D-02	1	0	5	4.6D-02	4.60D-02	1.D 01	4.D 00	1.D 00	FTTF
5	1	1.0D 00	7	1.1395D-02	1	0	5	3.2D-02	3.17D-02	9.D 01	2.D 01	1.D 00	FTTF
6	2	1.0D 00	8	5.6521D-03	0	0	6	1.3D-01	1.34D-01	1.D 02	1.D 01	1.D 00	FTTF
7	1	1.0D 00	9	4.8169D-03	0	0	6	4.0D-02	4.05D-02	9.D 01	1.D 01	1.D 00	FTTF
8	1	1.0D 00	10	4.5829D-03	0	0	6	1.4D-02	1.40D-02	8.D 01	9.D 00	1.D 00	FTTF
9	1	1.0D 00	11	4.4447D-03	0	0	6	1.5D-02	1.48D-02	1.D 02	2.D 01	1.D 00	FTTF
27	1	1.0D 00	29	1.1263D-07	0	0	6	6.2D-04	6.16D-04	7.D 03	7.D 02	1.D 00	FTTF
28	1	1.0D 00	30	7.8278D-09	0	0	6	8.1D-05	8.08D-05	6.D 03	7.D 02	1.D 00	FTTF
29	1	1.0D 00	31	1.8175D-09	0	0	6	3.7D-05	3.66D-05	6.D 03	8.D 02	1.D 00	FTTF
30	1	1.0D 00	32	1.1704D-09	0	0	6	1.8D-05	1.76D-05	7.D 03	7.D 02	1.D 00	FTTF
31	1	1.0D 00	33	1.1161D-09	0	0	6	4.3D-06	4.31D-06	7.D 03	7.D 02	1.D 00	TTTF
32	1	1.0D 00	34	1.1144D-09	0	0	6	4.0D-07	3.95D-07	7.D 03	7.D 02	1.D 00	TTTT

EXIT NP PHASE. INFORM = 0 MAJITS = 32 NFEVAL = 34

Figure 2a. Results of an IQP method on the six-variable problem Exp 6.

ITN	JDEL	JADD	STEP	NUMF	OBJECTIVE	BND	LC	NZ	NORM GZ	MIN LM	COND HZ	COND T	CONV
0	0	0	0.0D-01	1	1.8581D-01	4	0	2	8.17D-01	--	1.D 00	1.D 00	F FFF
1	0	0	1.3D-01	3	1.4147D-01	4	0	2	5.02D-01	--	1.D 00	1.D 00	T FTF
2	0	0	1.0D 00	4	2.3549D-02	4	0	2	4.81D-02	--	1.D 00	1.D 00	T FTF
3	6L	0	3.3D-01	6	2.3056D-02	3	0	3	2.56D-02	-5.2D-02	2.D 00	1.D 00	T FTF
4	5L	0	1.0D 00	7	2.1942D-02	2	0	4	7.82D-02	-4.0D-02	1.D 01	1.D 00	F FTF
5	0	0	1.0D 00	8	1.9597D-02	2	0	4	1.14D-01	--	1.D 02	1.D 00	F FTF
6	0	0	1.0D 00	9	1.2396D-02	2	0	4	3.46D-01	--	6.D 01	1.D 00	F FTF
7	0	0	1.0D 00	10	8.1884D-03	2	0	4	1.28D-01	--	4.D 01	1.D 00	F FTF
8	0	0	1.0D 00	11	6.7824D-03	2	0	4	6.57D-02	--	4.D 01	1.D 00	F FTF
9	0	0	1.0D 00	12	6.1975D-03	2	0	4	4.12D-03	--	3.D 01	1.D 00	T FTF
10	4L	0	1.0D 00	13	6.1445D-03	1	0	5	9.28D-03	-7.1D-03	2.D 01	1.D 00	F FTF
11	0	0	1.0D 00	14	5.9625D-03	1	0	5	1.25D-02	--	6.D 01	1.D 00	F FTF
12	0	0	1.0D 00	15	5.5139D-03	1	0	5	2.17D-02	--	2.D 02	1.D 00	F FTF
35	0	0	1.0D 00	39	1.1179D-05	1	0	5	3.10D-03	--	8.D 03	1.D 00	F FTF
36	0	0	1.0D 00	40	1.0212D-05	1	0	5	1.14D-03	--	9.D 03	1.D 00	F FTF
37	0	SU	8.3D-01	41	9.8389D-06	2	0	4	5.67D-04	--	4.D 02	1.D 00	F FTF
38	0	0	1.0D 00	42	9.8011D-06	2	0	4	4.54D-04	--	5.D 02	1.D 00	F FTF
39	0	0	1.0D 00	43	9.7912D-06	2	0	4	3.67D-05	--	5.D 02	1.D 00	F FTF
40	0	0	1.0D 00	44	9.7911D-06	2	0	4	1.02D-06	--	5.D 02	1.D 00	F FTT
41	0	0	1.0D 00	45	9.7911D-06	2	0	4	5.64D-07	--	6.D 02	1.D 00	F FTT
42	0	0	1.0D 00	46	9.7911D-06	2	0	4	1.21D-07	--	6.D 02	1.D 00	T FTT
43	2L	0	1.0D 00	47	9.7906D-06	1	0	5	3.64D-05	-2.4D-05	9.D 02	1.D 00	F FTT
44	0	0	1.0D 00	48	9.7878D-06	1	0	5	1.25D-04	--	1.D 03	1.D 00	F FTF
45	0	0	1.0D 00	49	9.7798D-06	1	0	5	2.90D-04	--	1.D 03	1.D 00	F FTF
53	0	0	1.0D 00	57	8.9619D-06	1	0	5	7.41D-05	--	1.D 04	1.D 00	F FTF
54	0	0	1.0D 00	58	8.9613D-06	1	0	5	6.24D-06	--	1.D 04	1.D 00	T FTF
55	SU	0	1.0D 00	59	8.9611D-06	0	0	6	3.61D-05	-1.4D-05	1.D 04	1.D 00	F FTT
116	0	0	1.0D 00	140	4.5834D-10	0	0	6	7.09D-05	--	1.D 05	1.D 00	F FTF
117	0	0	1.0D 00	141	5.7971D-11	0	0	6	5.01D-06	--	7.D 04	1.D 00	T TTF
118	0	0	1.0D 00	142	1.4296D-11	0	0	6	3.31D-06	--	1.D 05	1.D 00	T TTF
119	0	0	1.0D 00	143	1.5776D-12	0	0	6	2.44D-06	--	1.D 05	1.D 00	T TTF
120	0	0	1.0D 00	144	5.2595D-13	0	0	6	1.07D-06	--	8.D 04	1.D 00	T TTF
121	0	0	1.0D 00	145	1.1712D-14	0	0	6	1.52D-07	--	9.D 04	1.D 00	T TTT

EXIT LC PHASE. INFORM = 0 ITER = 121 NFEVAL = 145

Figure 2b. Results of an EQP method on the six-variable problem Exp 6.

ITN	ITQP	STEP	NUMF	OBJECTIVE	BND	LC	NZ	NORM GF	NORM GZ	COND HZ	COND N	COND T	CONV
0	56	0.0	1	-2.28390D+02	25	3	72	5.90D+01	5.41D+01	2.0D+01	1.0D+00	1.0D+00	FFFT
1	10	1.00D+00	2	-1.18920D+03	16	3	81	1.90D+01	1.84D+01	3.0D+01	2.0D+00	1.0D+00	FTFF
2	3	1.00D+00	3	-1.43820D+03	16	3	81	9.20D+00	9.16D+00	2.0D+01	2.0D+00	1.0D+00	FFFF
3	15	1.00D+00	4	-1.56730D+03	29	4	67	5.70D+00	5.56D+00	2.0D+01	2.0D+00	1.0D+00	FFFF
4	3	1.00D+00	5	-1.64010D+03	27	4	69	3.70D+00	3.43D+00	4.0D+01	3.0D+00	1.0D+00	FTTF
5	4	1.00D+00	6	-1.66310D+03	29	5	66	2.60D+00	2.36D+00	3.0D+01	4.0D+00	5.0D+00	FFFF
6	2	1.00D+00	7	-1.67960D+03	28	5	67	2.00D+00	1.77D+00	1.0D+02	4.0D+00	1.0D+00	FTFF
7	8	1.00D+00	8	-1.68060D+03	34	6	60	1.50D+00	1.35D+00	2.0D+02	4.0D+00	1.0D+00	FFFF
8	2	1.00D+00	9	-1.69150D+03	35	6	59	1.40D+00	1.22D+00	2.0D+02	4.0D+00	1.0D+00	FFFFS
9	2	1.00D+00	10	-1.69310D+03	36	6	58	1.30D+00	1.12D+00	2.0D+02	4.0D+00	1.0D+00	FFFFS
10	1	1.00D+00	11	-1.69450D+03	36	6	58	1.30D+00	1.06D+00	2.0D+02	4.0D+00	1.0D+00	FTFFS
11	2	2.70D+00	13	-1.69780D+03	37	6	57	1.10D+00	9.54D-01	2.0D+02	4.0D+00	1.0D+00	FTFFS
12	2	1.00D+00	15	-1.69890D+03	38	6	56	1.10D+00	9.21D-01	2.0D+02	4.0D+00	1.0D+00	FTFFS
13	3	1.20D+00	17	-1.70010D+03	38	6	56	1.10D+00	8.92D-01	2.0D+02	4.0D+00	1.0D+00	FTFFS
14	2	1.70D+00	19	-1.70160D+03	39	6	55	1.10D+00	8.43D-01	2.0D+02	4.0D+00	1.0D+00	FTFFS
15	2	4.10D+00	21	-1.70490D+03	40	6	54	8.90D-01	6.61D-01	7.0D+01	4.0D+00	1.0D+00	FTFFS
16	2	3.60D+00	23	-1.70710D+03	41	6	53	8.50D-01	6.26D-01	9.0D+01	4.0D+00	1.0D+00	FTFFS
17	2	3.60D+00	25	-1.70860D+03	42	6	52	8.40D-01	6.25D-01	9.0D+01	4.0D+00	1.0D+00	FTFFS
18	2	6.80D+00	28	-1.71130D+03	43	6	51	7.80D-01	5.77D-01	9.0D+01	4.0D+00	1.0D+00	FTFFS
19	2	6.90D+00	31	-1.71330D+03	44	6	50	7.40D-01	5.49D-01	9.0D+01	4.0D+00	1.0D+00	FTFFS
20	2	1.90D+00	33	-1.71370D+03	45	6	49	7.30D-01	5.37D-01	9.0D+01	4.0D+00	1.0D+00	FTFFS
30	1	1.00D+00	53	-1.72610D+03	56	6	38	4.60D-01	2.42D-01	6.0D+01	3.0D+01	1.0D+00	FTFFS
40	3	6.00D+00	73	-1.73230D+03	62	7	31	3.40D-01	1.37D-01	2.0D+02	8.0D+01	2.0D+00	FFFF
50	2	1.00D+00	84	-1.73310D+03	63	7	30	3.40D-01	1.34D-01	1.0D+03	1.0D+02	2.0D+00	FTFF
60	1	1.00D+00	94	-1.73370D+03	64	7	29	2.80D-01	7.20D-02	5.0D+02	1.0D+02	1.0D+00	FTFF
70	5	1.00D+00	104	-1.73400D+03	65	7	28	2.90D-01	9.86D-02	6.0D+02	1.0D+02	1.0D+00	FFFF
80	1	1.00D+00	114	-1.73450D+03	69	7	24	2.70-01	6.03D-02	1.0D+02	9.0D+01	1.0D+00	FTFF
90	1	1.00D+00	124	-1.73460D+03	69	7	24	2.60-01	3.46D-02	6.0D+02	1.0D+02	1.0D+00	FTFF
100	1	1.00D+00	134	-1.73460D+03	70	7	23	2.60-01	2.03D-02	3.0D+02	1.0D+02	1.0D+00	FTFF
110	1	1.00D+00	144	-1.73460D+03	71	7	22	2.60-01	1.55D-02	2.0D+02	1.0D+02	1.0D+00	TTFF
120	1	1.00D+00	154	-1.73470D+03	72	7	21	2.50-01	2.46D-02	5.0D+02	1.0D+02	2.0D+00	FTFF
130	1	1.00D+00	164	-1.73470D+03	72	7	21	2.50-01	2.09D-02	9.0D+02	1.0D+02	2.0D+00	FTFF
140	1	1.00D+00	174	-1.73470D+03	72	7	21	2.70-01	2.07D-02	8.0D+03	2.0D+02	1.0D+00	FTFF
150	1	1.00D+00	184	-1.73490D+03	73	7	20	2.60-01	3.39D-02	2.0D+03	2.0D+03	1.0D+00	FTFF
160	1	1.00D+00	194	-1.73490D+03	73	7	20	2.60-01	3.59D-02	2.0D+03	3.0D+03	1.0D+00	FTFF
170	2	1.00D+00	204	-1.73500D+03	75	7	18	2.50-01	2.18D-02	8.0D+02	5.0D+03	2.0D+00	FFFF
180	1	1.00D+00	214	-1.73500D+03	75	7	18	2.50-01	9.33D-03	8.0D+02	5.0D+03	2.0D+00	TTFF
190	1	1.00D+00	224	-1.73500D+03	75	7	18	2.50-01	3.87D-03	6.0D+02	5.0D+03	2.0D+00	TTFF
200	1	1.00D+00	234	-1.73500D+03	75	7	18	2.50-01	1.62D-03	5.0D+02	5.0D+03	2.0D+00	TTFF
210	1	1.00D+00	244	-1.73500D+03	75	7	18	2.50-01	2.47D-04	5.0D+02	5.0D+03	2.0D+00	TTFF
211	1	1.00D+00	245	-1.73500D+03	75	7	18	2.50-01	1.31D-04	5.0D+02	5.0D+03	2.0D+00	TTFF
212	1	1.00D+00	246	-1.73500D+03	75	7	18	2.50-01	6.18D-05	5.0D+02	5.0D+03	2.0D+00	TTFF
213	1	1.00D+00	247	-1.73500D+03	75	7	18	2.50-01	4.90D-05	5.0D+02	5.0D+03	2.0D+00	TTFF
214	1	1.00D+00	248	-1.73500D+03	75	7	18	2.50-01	4.78D-05	5.0D+02	5.0D+03	2.0D+00	TTFT

EXIT NP PHASE. INFORM = 0 MAJITS = 214 NFEVAL = 248

Figure 3a. Results of an IQP method on the Weapon problem.

two major iterations. During the early QP subproblems the poor Hessian approximation causes the IQP method to delete too many constraints from the working set. Further minor iterations are then needed to add the constraints that were deleted unnecessarily. The large dimension of the projected Hessian also inhibits the ability of the method to recover quickly. Weapon has a Hessian matrix with many negative and zero eigenvalues. If there are not enough constraints in the working set, the linesearch function is likely to have negative curvature along the search direction and the quasi-Newton update will be skipped. If this occurs for several consecutive iterations, the original poor curvature information is left unaltered – as is the inability of the QP to predict the correct active set. This behavior was apparent during iterations 8–30 where no quasi-Newton update could be performed.

ITN	JDEL	JADD	STEP	NUMF	OBJECTIVE	BND	LC	NZ	NORM GZ	MIN LM	COND HZ	COND T	CONV
0	0	0	0.00-01	1	-1.95390 02	91	9	0	0.000-01	--	1.D 00	4.D 00	T TFF
1	72L	0	1.00 00	2	-3.15650 02	90	9	1	5.280 00	-2.70 01	1.D 00	4.D 00	T FTF
2	101U	0	1.00 00	3	-3.63720 02	90	8	2	4.440 00	-2.40 01	1.D 00	1.D 00	T FTF
3	67L	0	1.00 00	4	-5.24900 02	89	8	3	5.470 00	-2.20 01	2.D 00	1.D 00	T FTF
4	97L	0	1.00 00	5	-6.84580 02	88	8	4	4.600 00	-1.90 01	2.D 00	2.D 00	T FTF
5	87L	0	1.00 00	6	-8.12370 02	87	8	5	3.430 00	-1.30 01	3.D 00	2.D 00	T FTF
6	77L	96L	2.10-01	7	-8.42750 02	87	8	5	7.900 00	-1.30 01	5.D 00	2.D 00	T FTF
7	64L	46L	1.40-01	8	-8.79370 02	87	8	5	1.210 01	-1.20 01	3.D 00	2.D 00	T FTF
8	0	0	1.00 00	9	-9.94380 02	87	8	5	5.280 00	--	3.D 00	2.D 00	T FTF
9	82L	0	1.00 00	10	-1.10430 03	86	8	6	3.200 00	-1.30 01	3.D 00	3.D 00	T FTF
10	92L	0	1.00 00	11	-1.20140 03	85	8	7	2.530 00	-1.30 01	3.D 00	3.D 00	T FTF
20	22L	0	1.00 00	21	-1.50830 03	78	8	14	1.420 00	-3.50 00	6.D 00	3.D 00	T FTF
30	0	0	1.00 00	31	-1.58920 03	75	8	17	1.510 00	--	3.D 01	3.D 00	T FTF
40	0	87L	5.50-01	41	-1.66140 03	78	7	15	1.110 00	--	1.D 01	3.D 00	T FTF
50	0	0	1.00 00	51	-1.69350 03	77	6	17	1.080 00	--	1.D 01	3.D 00	T FTF
60	0	0	1.00 00	62	-1.71000 03	79	7	14	4.360-01	--	4.D 01	2.D 00	T FTF
70	0	0	1.00 00	72	-1.71520 03	78	8	14	3.360-01	--	3.D 01	2.D 00	T FTF
80	0	0	1.00 00	82	-1.71900 03	78	8	14	2.490-01	--	6.D 01	2.D 00	T FTF
90	0	0	1.00 00	92	-1.72420 03	78	8	14	2.050-01	--	3.D 02	2.D 00	T FTF
100	0	0	1.00 00	102	-1.72670 03	77	8	15	2.740-01	--	2.D 01	2.D 00	T FTF
110	0	0	1.00 00	112	-1.72890 03	77	6	17	2.180-01	--	3.D 01	2.D 00	T FTF
120	0	0	1.00 00	122	-1.73090 03	78	6	16	1.090-01	--	4.D 01	2.D 00	T FTF
130	0	0	1.00 00	132	-1.73150 03	77	6	17	1.690-01	--	3.D 01	2.D 00	T FTF
140	0	0	1.00 00	142	-1.73220 03	79	6	15	6.960-02	--	1.D 02	2.D 00	T FTF
150	0	0	1.00 00	152	-1.73260 03	78	6	16	5.980-02	--	2.D 02	2.D 00	T FTF
160	0	0	1.00 00	162	-1.73290 03	78	6	16	3.990-02	--	7.D 01	2.D 00	T FTF
170	0	0	1.00 00	172	-1.73310 03	77	6	17	3.220-02	--	8.D 01	2.D 00	T FTF
180	0	0	1.00 00	182	-1.73330 03	76	6	18	6.320-02	--	9.D 01	2.D 00	T FTF
190	73U	0	1.00 00	192	-1.73350 03	75	6	19	2.270-02	-2.00-02	2.D 02	2.D 00	T FTF
200	0	0	1.00 00	202	-1.73370 03	75	6	19	4.690-02	--	2.D 02	2.D 00	T FTF
210	0	0	1.00 00	212	-1.73440 03	75	7	18	5.690-02	--	2.D 01	2.D 00	T FTF
220	0	0	1.00 00	222	-1.73470 03	75	7	18	2.360-02	--	3.D 01	2.D 00	T FTF
230	0	0	1.00 00	232	-1.73480 03	74	7	19	6.150-03	--	3.D 01	2.D 00	T FTF
240	0	0	1.00 00	242	-1.73500 03	75	7	18	1.760-04	--	4.D 01	2.D 00	T TTF
241	0	0	1.00 00	243	-1.73500 03	75	7	18	5.820-05	--	4.D 01	2.D 00	T TTF
242	0	0	1.00 00	244	-1.73500 03	75	7	18	1.450-05	--	4.D 01	2.D 00	T TTF
243	0	0	1.00 00	245	-1.73500 03	75	7	18	1.930-06	--	4.D 01	2.D 00	T TTT

EXIT LC PHASE. INFORM = 0 ITER = 243 NFEVAL = 245

Figure 3b. Results of an EQP method on the Weapon problem.

It is difficult to predict in advance whether a particular problem will be more suitable for an EQP method or an IQP method. IQP methods are likely to be less efficient on problems for which the QP multipliers change rapidly from one iteration to the next. Problems in this category tend to be highly nonlinear or to have many small Lagrange multipliers. In either case, the significant changes to the working set between iterations seriously impair the ability of the quasi-Newton update to build useful curvature information about the function. Conversely, IQP methods will tend to be efficient if the QP multiplier estimates are very accurate when computed at points that are far from the solution (for example, if the problem were close to being quadratic).

To a large extent, the relative efficiency of IQP and EQP methods depends upon the number of constraints active at the solution. EQP methods are usually implemented so that as many constraints as possible are included in the initial working set. It is therefore not surprising that they tend to be more efficient when more constraints are active at the solution. Finally, the relative efficiency of a method is critically dependent upon the ratio of the amount of work required to perform a single minor iteration compared to the work required to evaluate the problem functions. As this ratio increases (as it often does as the size of the problem increases), the advantage will

swing towards the EQP method.

5.3. Typical performance of an SQP method for nonlinear constraints. The remaining runs were obtained from Version 2.1 of the program NPSOL (see Gill *et al.*, 1984b), an IQP quasi-Newton method for nonlinearly constrained optimization. The Hessian of each QP subproblem is a positive-definite BFGS approximation to the Hessian of the Lagrangian function. The QP subproblem is solved using a feasible-point active-set method with an orthogonal factorization of the constraints in the working set.

The merit function used in NPSOL is a smooth augmented Lagrangian function that utilizes the properties of *slack variables*. The inequality constraints of NIP can be reformulated as equality constraints by adding simply-bounded slack variables s_j. Estimates of the slack variables are used in the linesearch to give a smooth augmented Lagrangian function. At each major iteration, a vector triple $(p, \delta\lambda, \delta s)$ is computed that serves as a direction of search for the variables x, multiplier estimates λ, and slack variables s. (All the elements of the vector triple are available from the solution of the standard IQP subproblem considered in Section 4.2. The vector $\delta\lambda$ is defined as $\mu - \lambda$, where μ are the QP multipliers, and the vector δs is given by $Ap + c - s$. Note that the QP solver does *not* need to treat the elements of s as additional variables.) The steplength is required to produce a sufficient decrease in the augmented Lagrangian merit function

$$\mathcal{L}(x, \lambda, s) = F(x) - \sum_{i=1}^{m} \lambda_i \big(c_i(x) - s_i\big) + \frac{\rho}{2} \sum_{i=1}^{m} \big(c_i(x) - s_i\big)^2.$$

The value of ρ is initially set to zero, and is occasionally increased from its value in the previous iteration in order to ensure descent for the merit function. Thus the sequence of penalty parameters is generally non-decreasing, although NPSOL has the ability to reduce the value of the penalty parameter a limited number of times.

The iteration summary printed in each of Figures 4.6 is identical to that provided by the linearly constrained IQP method, except that the merit function value ("MERIT") is printed instead of the objective value, and the additional columns "NC", "NORM C" and "RHO" give the number of nonlinear constraints in the working set, the two-norm of the residuals of constraints in the working set and the penalty parameter used in the merit function. In all of the NPSOL runs, the feasibility tolerance for each nonlinear constraint was set at 10^{-6}.

Two runs were selected to illustrate the behavior of an IQP method when solving well behaved (but non-trivial) nonlinear problems. Figure 4 gives the results obtained on a version of the Hexagon problem. (For more details of this problem, see Wright, 1976. A slightly different formulation is given as Problem 108 by Hock and Schittkowski, 1981.) Hexagon is a popular test problem for nonlinear programming methods. All constraint types are included (bounds, linear, nonlinear), and the Hessian of the Lagrangian function is not positive definite at the solution. The problem has nine variables, finite bounds on six of the variables, four general linear constraints, and fifteen nonlinear constraints. Six nonlinear constraints are active at x^*.

The problem solved in Figure 5 is derived from a 30-bus optimal power flow (OPF) problem of optimizing the distribution of electrical power over a network. The problem has 67 variables, 60 nonlinear constraints, and upper and lower bounds on all of the variables. At the solution, 54 nonlinear constraints and three simple bounds are active.

ITN	ITQP	STEP	NUMF	MERIT	BND	LC	NC	NZ	NORM GF	NORM GZ	COND HZ	COND H	COND T	NORM C	RHO	CONV
0	6	0.0D-01	1	-1.4333D 00	0	0	5	4	2.0D 00	2.28D-01	1.D 00	1.D 00	3.D 00	1.15D 00	0.0D-01	FFTT
1	8	4.0D-01	3	-1.3952D 00	0	0	4	5	2.1D 00	9.21D-02	1.D 00	2.D 00	2.D 00	6.15D-01	5.3D-01	FFTF
2	1	1.0D 00	4	-1.2995D 00	0	0	4	5	2.0D 00	1.64D-01	2.D 00	2.D 00	1.D 00	6.60D-02	1.1D 00	FFTF
3	3	1.0D 00	5	-1.3222D 00	0	0	6	3	2.0D 00	1.88D-01	1.D 00	7.D 00	2.D 00	2.06D-01	1.1D 00	FFTF
4	1	1.0D 00	6	-1.3463D 00	0	0	6	3	2.1D 00	1.19D-01	2.D 00	4.D 02	2.D 00	1.40D-02	1.1D 00	FFTF
5	1	1.0D 00	7	-1.3496D 00	0	0	6	3	2.1D 00	2.74D-02	1.D 00	3.D 02	2.D 00	4.95D-03	1.1D 00	FFTF
6	1	1.0D 00	8	-1.3498D 00	0	0	6	3	2.1D 00	1.02D-02	2.D 00	2.D 02	2.D 00	1.12D-03	1.1D 00	FFTF
7	1	1.0D 00	9	-1.3499D 00	0	0	6	3	2.1D 00	8.10D-03	4.D 00	3.D 02	2.D 00	9.79D-05	1.1D 00	FFTF
8	1	1.0D 00	10	-1.3500D 00	0	0	6	3	2.1D 00	1.23D-03	4.D 00	2.D 02	2.D 00	5.00D-04	1.1D 00	FFTF
9	1	1.0D 00	11	-1.3500D 00	0	0	6	3	2.1D 00	9.35D-05	4.D 00	2.D 02	2.D 00	1.13D-06	1.4D 01	FTTF
10	1	1.0D 00	12	-1.3500D 00	0	0	6	3	2.1D 00	2.37D-05	4.D 00	2.D 02	2.D 00	5.90D-09	5.6D 00	FTTF
11	1	1.0D 00	13	-1.3500D 00	0	0	6	3	2.1D 00	2.35D-06	4.D 00	2.D 02	2.D 00	4.52D-10	5.6D 00	TTTT

EXIT NP PHASE. INFORM = 0 MAJITS = 11 NFEVAL = 13 NCEVAL = 13

VARIABLE	STATE	VALUE	LOWER BOUND	UPPER BOUND	LAGR MULTIPLIER	RESIDUAL
VARBL 1	FR	0.6094650D-01	0.0000000	NONE	0.0000000	0.6095D-01
VARBL 2	FR	0.5976502	NONE	NONE	0.0000000	0.1000D 11
VARBL 3	FR	1.000000	NONE	NONE	0.0000000	0.1000D 11
VARBL 4	FR	0.5976497	NONE	NONE	0.0000000	0.1000D 11
VARBL 5	FR	0.6094641D-01	0.0000000	NONE	0.0000000	0.6095D-01
VARBL 6	FR	0.3437710	0.0000000	NONE	0.0000000	0.3438
VARBL 7	FR	0.5000000	0.0000000	NONE	0.0000000	0.5000
VARBL 8	FR	-0.5000000	NONE	0.0000000	0.0000000	0.5000
VARBL 9	FR	-0.3437708	NONE	0.0000000	0.0000000	0.3438

LINEAR CONSTR	STATE	VALUE	LOWER BOUND	UPPER BOUND	LAGR MULTIPLIER	RESIDUAL
LNCON 1	FR	-0.5367037	NONE	0.0000000	0.0000000	0.5367
LNCON 2	FR	-0.4023498	NONE	0.0000000	0.0000000	0.4023
LNCON 3	FR	0.4023503	0.0000000	NONE	0.0000000	0.4024
LNCON 4	FR	0.5367033	0.0000000	NONE	0.0000000	0.5367

NONLNR CONSTR	STATE	VALUE	LOWER BOUND	UPPER BOUND	LAGR MULTIPLIER	RESIDUAL
NLCON 1	FR	0.8781070	0.0000000	NONE	0.0000000	0.8781
NLCON 2	FR	0.6875417	0.0000000	NONE	0.0000000	0.6875
NLCON 3	LL	-0.4990394D-10	0.0000000	NONE	0.8318393D-01	-0.4990D-10
NLCON 4	LL	-0.3452747D-09	0.0000000	NONE	0.3202626	-0.3453D-09
NLCON 5	FR	0.5272863	0.0000000	NONE	0.0000000	0.5273
NLCON 6	FR	0.3928143	0.0000000	NONE	0.0000000	0.3928
NLCON 7	FR	0.5881147	0.0000000	NONE	0.0000000	0.5881
NLCON 8	LL	-0.2877292D-09	0.0000000	NONE	0.1992985	-0.2877D-09
NLCON 9	LL	-0.1479264D-10	0.0000000	NONE	0.3202626	-0.1479D-10
NLCON 10	UL	0.0000000	NONE	0.0000000	-0.3437718	0.0000
NLCON 11	FR	0.5881142	0.0000000	NONE	0.0000000	0.5881
NLCON 12	LL	-0.1845829D-11	0.0000000	NONE	0.8318383D-01	-0.1846D-11
NLCON 13	FR	0.3928147	0.0000000	NONE	0.0000000	0.3928
NLCON 14	FR	0.6875419	0.0000000	NONE	0.0000000	0.6875
NLCON 15	FR	0.8781072	0.0000000	NONE	0.0000000	0.8781

EXIT NPSOL - OPTIMAL SOLUTION FOUND.

Figure 4. Output from the solution of the well-behaved problem Hexagon.

On these two well behaved problems, the approximate Hessian and working set remain relatively well-conditioned. Similarly, the penalty parameters remain small and approximately con-

stant. The two runs illustrate much of the numerical behavior of a quasi-Newton IQP method that is predicted from theoretical analysis. As x_k approaches the solution, just one minor iteration is performed per major iteration, and entries in the "NORM GZ" and "NORM C" columns exhibit the superlinear convergence rate discussed in Section 3.3. Note that the constraint violations converge earlier than the projected gradient. The final values of the projected gradient norm and constraint norm reflect the limiting accuracy of the two quantities. It is possible to achieve almost full precision in the constraint norm but only half precision in the projected-gradient norm.

ITN	ITQP	STEP	NUMF	MERIT	BND	LC	NC	NZ	NORM GF	NORM GZ	COND HZ	COND H	COND T	NORM C	RHO	CONV
0	20	0.0	1	-1.09110D+00	1	0	59	7	1.00+00	7.73D-14	1.D+00	1.D+00	0.D+01	2.92D+00	0.0	TFFT
1	0	7.0D-01	3	1.00100D+00	1	0	55	11	1.00+00	4.54D-01	2.D+00	3.D+00	9.D+01	1.67D+00	4.0D+00	FFTF
2	6	1.00+00	4	1.00430D+00	1	0	56	10	1.00+00	2.24D-01	7.D+00	6.D+00	8.D+01	6.31D-01	4.0D+00	FFTF
3	2	4.9D-01	6	9.91420-01	1	0	56	10	1.00+00	4.35D-02	1.D+01	1.D+01	9.D+01	7.12D-01	2.0D+00	FFTF
4	3	4.3D-01	8	9.88350-01	1	0	54	12	1.00+00	6.08D-02	9.D+00	1.D+01	9.D+01	1.65D-02	2.0D+00	FFTF
5	1	1.00+00	9	9.87690-01	1	0	54	12	1.00+00	7.09D-02	2.D+01	3.D+01	9.D+01	1.09D-02	2.0D+00	FFTF
6	1	1.00+00	10	9.86420-01	1	0	54	12	1.00+00	3.35D-02	3.D+01	3.D+01	9.D+01	7.49D-04	2.0D+00	FFTF
7	1	1.00+00	11	9.84500-01	1	0	54	12	1.00+00	4.22D-02	6.D+01	3.D+01	9.D+01	3.84D-03	2.0D+00	FFTF
8	4	1.00+00	12	9.81540-01	3	0	55	9	1.00+00	7.38D-02	8.D+01	4.D+01	9.D+01	1.18D-01	2.0D+00	FFFF
9	4	1.00+00	13	9.77670-01	5	0	54	8	1.00+00	7.92D-02	8.D+00	9.D+01	8.D+01	4.29D-02	2.0D+00	FFFF
10	2	1.00+00	14	9.75840-01	6	0	54	7	1.00+00	4.22D-02	8.D+00	9.D+01	9.D+01	4.59D-03	2.0D+00	FFTF
11	1	1.00+00	15	9.74610-01	6	0	54	7	1.00+00	1.75D-02	2.D+01	9.D+01	9.D+01	6.48D-04	2.0D+00	FFTF
12	3	1.00+00	16	9.74310-01	4	0	54	9	1.00+00	1.90D-02	2.D+01	9.D+01	9.D+01	1.78D-04	2.0D+00	FFFF
13	2	1.00+00	17	9.74150-01	3	0	54	10	1.00+00	1.37D-02	3.D+01	9.D+01	9.D+01	9.68D-05	2.0D+00	FFTF
14	2	1.00+00	18	9.74030-01	2	0	54	11	1.00+00	9.67D-03	5.D+01	8.D+01	9.D+01	1.36D-04	2.0D+00	FFTF
15	1	1.00+00	19	9.73920-01	2	0	54	11	1.00+00	9.20D-03	9.D+01	8.D+01	9.D+01	1.99D-04	2.0D+00	FFTF
16	2	1.00+00	20	9.73740-01	2	0	55	10	1.00+00	1.67D-02	7.D+01	1.D+02	9.D+01	3.38D-02	2.0D+00	FFFF
17	3	1.00+00	21	9.73440-01	3	0	54	10	1.00+00	1.25D-02	3.D+01	3.D+02	9.D+01	8.56D-03	2.0D+00	FFFF
18	1	1.00+00	22	9.73080-01	3	0	54	10	1.00+00	1.21D-02	1.D+02	7.D+02	1.D+02	1.99D-03	2.0D+00	FFFF
19	2	1.00+00	23	9.72670-01	4	0	54	9	1.00+00	1.16D-02	3.D+01	9.D+02	1.D+02	5.01D-03	2.0D+00	FFFF
20	1	1.00+00	24	9.72540-01	4	0	54	9	1.00+00	6.74D-03	5.D+01	1.D+03	1.D+02	8.28D-05	2.0D+00	FFFF
21	1	1.00+00	25	9.72390-01	4	0	54	9	1.00+00	5.39D-03	7.D+01	1.D+03	1.D+02	7.99D-04	2.0D+00	FFTF
22	1	1.00+00	26	9.72360-01	4	0	54	9	1.00+00	5.18D-03	2.D+02	1.D+03	1.D+02	8.66D-05	2.0D+00	FFTF
23	2	1.00+00	27	9.72210-01	3	0	54	10	1.00+00	6.83D-03	3.D+02	1.D+03	1.D+02	4.21D-03	2.0D+00	FFTF
24	1	1.00+00	28	9.72130-01	3	0	54	10	1.00+00	6.81D-03	4.D+02	1.D+03	1.D+02	2.33D-03	2.0D+00	FFTF
25	2	1.00+00	29	9.72090-01	4	0	54	9	1.00+00	3.43D-03	1.D+02	9.D+02	1.D+02	1.46D-03	2.0D+00	FFTF
26	1	1.00+00	30	9.72070-01	4	0	54	9	1.00+00	1.86D-03	1.D+02	8.D+02	1.D+02	8.32D-05	2.0D+00	FFTF
27	2	1.00+00	31	9.72070-01	3	0	54	10	1.00+00	2.26D-03	1.D+02	8.D+02	1.D+02	2.99D-05	2.0D+00	FFTF
28	1	1.00+00	32	9.72060-01	3	0	54	10	1.00+00	2.53D-03	3.D+02	8.D+02	1.D+02	5.86D-05	2.0D+00	FFTF
29	2	1.00+00	33	9.72050-01	4	0	54	9	1.00+00	2.15D-03	2.D+02	8.D+02	1.D+02	1.06D-03	2.0D+00	FFTF
30	1	1.00+00	34	9.72050-01	4	0	54	9	1.00+00	1.33D-03	3.D+02	9.D+02	1.D+02	1.11D-04	2.0D+00	FFFF
31	2	1.00+00	35	9.72040-01	3	0	54	10	1.00+00	1.26D-03	2.D+02	9.D+02	1.D+02	5.17D-05	2.0D+00	FFTF
32	1	1.00+00	36	9.72040-01	3	0	54	10	1.00+00	8.55D-04	3.D+02	8.D+02	1.D+02	1.24D-05	2.0D+00	FFTF
33	1	1.00+00	37	9.72040-01	3	0	54	10	1.00+00	4.72D-04	3.D+02	6.D+02	1.D+02	2.68D-06	2.0D+00	FTTF
34	1	1.00+00	38	9.72040-01	3	0	54	10	1.00+00	1.89D-04	3.D+02	5.D+02	1.D+02	2.57D-06	2.0D+00	FTTF
35	1	1.00+00	39	9.72040-01	3	0	54	10	1.00+00	1.98D-04	3.D+02	5.D+02	1.D+02	3.25D-07	2.0D+00	FTTF
36	1	1.00+00	40	9.72040-01	3	0	54	10	1.00+00	2.28D-04	3.D+02	7.D+02	1.D+02	2.36D-07	2.0D+00	FTTF
37	1	1.00+00	41	9.72040-01	3	0	54	10	1.00+00	2.88D-04	4.D+02	9.D+02	1.D+02	1.14D-06	2.0D+00	FTTF
38	1	1.00+00	42	9.72040-01	3	0	54	10	1.00+00	3.19D-04	5.D+02	8.D+02	1.D+02	4.37D-06	2.0D+00	FTTF
39	1	1.00+00	43	9.72040-01	3	0	54	10	1.00+00	2.34D-04	5.D+02	7.D+02	1.D+02	6.05D-06	2.0D+00	FTTF
40	1	1.00+00	44	9.72040-01	3	0	54	10	1.00+00	6.35D-05	5.D+02	8.D+02	1.D+02	1.35D-06	2.0D+00	FTTF
41	1	1.00+00	45	9.72040-01	3	0	54	10	1.00+00	1.08D-05	5.D+02	8.D+02	1.D+02	6.87D-08	2.0D+00	TTTF
42	1	1.00+00	46	9.72040-01	3	0	54	10	1.00+00	6.66D-07	5.D+02	7.D+02	1.D+02	1.66D-08	2.0D+00	TTTF
43	1	1.00+00	47	9.72040-01	3	0	54	10	1.00+00	8.78D-08	5.D+02	7.D+02	1.D+02	3.60D-10	2.0D+00	TTTT

EXIT NP PHASE. INFORM = 0 MAJITS = 43 NFEVAL = 47 NCEVAL = 47

Figure 5. Output from the solution of the OPF problem.

The status and values of the variables and constraints at the final solution give useful information about the progress of a minimization and the degree of difficulty of the problem. Figure 4 includes the final solution output from NPSOL for Hexagon. The printout is divided into three sections, giving information about the final status of the variables, general linear constraints and

nonlinear constraints, respectively. Within each section, "STATE" gives the status of the associated constraint in the predicted active set (FR if not included, EQ if a fixed value, LL if at its lower bound, and UL if at its upper bound). "VALUE" is the value of the constraint at the final iteration. "LOWER BOUND" and "UPPER BOUND" give the lower and upper bounds specified for the constraint ("NONE" indicates that no bound is enforced). "LAGR MULTIPLIER" is the value of the Lagrange multiplier. This will be zero if STATE is FR. The multiplier is non-negative if STATE is LL, and non-positive if STATE is UL. "RESIDUAL" gives the difference between the entry in the "VALUE" column and the nearer bound.

In the first section, "VARIABLE" is the name (VARBL) and index of a variable. In the linear constraints section, "LINEAR CONSTR" is the name (LNCON) and index of a linear constraint. "NONLNR CONSTR" is the name (NLCON) and index of a nonlinear constraint.

Note that, although the feasibility tolerance for the nonlinear constraints is of the order 10^{-6}, the final accuracy is considerably better than this. This is because the constraint violations are being refined during the last few iterations while the algorithm is working to reduce the projected-gradient norm. Another feature worth noting is that the the constraint values and Lagrange multipliers at the solution are "well balanced". For example, all the multipliers are approximately of the same order of magnitude. This behavior is typical of a well-scaled problem.

5.4. Performance on an ill-conditioned problem. Finally, we give the results of the IQP method on the problem Dembo 7. This problem is a geometric programming formulation developed by Dembo (1976) of a five-stage membrane separation process. The problem has sixteen variables, eight linear constraints, and eleven nonlinear constraints. All sixteen variables have simple upper and lower bound constraints. The problem causes many difficulties for a nonlinear programming algorithm because of bad scaling and linearly dependent constraints.

The results for Dembo 7 show a number of features that are common to badly behaved problems. First, note that the number of minor iterations does not decline quickly. Moreover, the presence of near-zero Lagrange multipliers sometimes causes the QP to require more than one iteration relatively close to the solution. A very common symptom of a badly behaved problem is the large value of the condition estimator of the full approximate Hessian, which is to be contrasted with the relatively modest value of the condition of the *projected* Hessian. This observation has some relevance to the choice of method for the QP subproblem. Clearly, special care must be taken when implementing any QP method that requires the factors of the full Hessian (as opposed to the projected Hessian). Starting the minor iterations at an unconstrained minimum of the QP subproblem will result in very large values of δp (see Section 4.3).

Note that the third bound constraint, the third linear constraint and eleventh nonlinear constraint all have very small residuals but are not in the working set. The values of the nonlinear constraints in the working set vary significantly in order of magnitude, indicating that the constraints are badly scaled. In constrast to the solution of Hexagon, in which the accuracy of the constraints was much better than required by the convergence tolerance, some of the nonlinear constraints are only just satisfied to the required feasibility tolerance.

ITN	ITQP	STEP	NUMF	MERIT	BND	LC	NC	NZ	NORM GF	NORM GZ	COND HZ	COND N	COND T	NORM C	RHO	CONV
0	30	0.0	1	2.8459D+02	1	3	9	3	1.1D+03	5.17D-01	1.D+00	1.D+00	7.D+03	2.19D-01	0.0	FFFT
1	13	9.6D-01	4	-2.1324D+02	3	2	5	6	8.1D+02	1.76D+00	4.D+01	6.D+06	2.D+02	1.50D+00	1.40+06	FFFF
2	12	6.1D-03	6	3.7305D+02	2	2	7	5	8.1D+02	1.87D+00	1.D+00	1.D+08	1.D+05	1.49D+00	1.6D+03	FFFF
3	12	5.7D-02	8	4.4162D+02	2	1	7	6	8.1D+02	1.12D+02	5.D+01	3.D+08	5.D+03	1.41D+00	8.00+02	FFFF
4	7	1.00+00	9	4.0200D+02	2	3	5	6	8.2D+02	2.25D+00	1.D+00	1.D+08	4.D+02	1.86D-01	8.00+02	FFFF
5	9	1.00+00	10	3.9278D+02	2	1	7	6	8.1D+02	1.62D+00	2.D+01	4.D+08	7.D+01	4.18D-01	1.3D+03	FFFF
6	17	1.00+00	11	3.0388D+02	2	3	5	6	1.7D+02	7.24D-01	3.D+08	2.D+10	4.D+03	3.03D-01	6.80+01	FFFF
7	20	1.00+00	12	1.3093D+02	4	2	8	2	1.5D+02	4.94D+00	8.D+01	7.D+11	1.D+05	9.00D-01	3.70+01	FFFF
8	11	5.00-01	14	-6.2190D+03	3	2	6	5	6.4D+02	3.12D+01	1.D+04	3.D+12	8.D+02	4.52D-01	6.40+03	FFFF
9	2	1.00+00	15	1.9803D+02	4	2	6	4	6.4D+02	3.21D+01	5.D+05	4.D+11	8.D+04	2.37D-02	3.80+02	FFFF
10	1	1.00+00	16	1.8996D+02	4	2	5	5	6.4D+02	3.19D+01	6.D+05	2.D+12	1.D+01	6.80D-06	3.60+03	FTFF
11	3	1.00+00	17	1.8987D+02	4	2	5	5	6.4D+02	3.17D+01	3.D+06	9.D+10	1.D+01	6.20D-07	6.2D+02	FTFF
12	6	1.00+00	18	1.8883D+02	6	1	5	4	1.6D+02	2.99D+01	1.D+02	5.D+11	1.D+01	1.09D-01	6.2D+02	FFFF
13	2	1.00+00	19	1.8809D+02	7	1	5	3	1.5D+02	2.29D-01	2.D+00	4.D+11	1.D+01	7.47D-03	6.2D+02	FFFF
14	4	1.00+00	20	1.8357D+02	5	2	6	3	6.3D+02	2.35D-01	2.D+00	1.D+12	2.D+01	1.97D-03	6.2D+02	FFFF
15	3	1.00+00	21	1.8352D+02	5	2	6	3	1.3D+02	2.36D-01	4.D+01	5.D+11	3.D+03	3.04D-06	1.1D+05	FTFF
16	4	1.00+00	22	1.7857D+02	4	2	8	2	6.2D+02	2.41D-02	1.D+00	1.D+12	8.D+03	1.44D-02	6.50+03	FFFF
17	1	1.00+00	23	1.7509D+02	4	2	8	2	6.2D+02	6.76D-02	1.D+00	1.D+10	7.D+03	3.04D-03	2.90+03	FFFF
18	2	1.00+00	24	1.7506D+02	5	2	7	2	6.2D+02	6.25D-02	1.D+00	1.D+10	7.D+03	4.87D-05	2.90+03	FFFF
19	8	1.00+00	25	1.7505D+02	6	2	6	2	6.2D+02	1.05D+00	2.D+01	2.D+12	7.D+02	1.56D-02	2.90+03	FFFF
20	2	1.00+00	26	1.7479D+02	6	2	7	1	6.2D+02	8.91D-03	1.D+00	1.D+11	8.D+02	1.02D-03	2.90+03	FFFF
21	1	1.00+00	27	1.7480D+02	6	2	7	1	6.2D+02	8.54D-03	1.D+00	7.D+10	8.D+02	1.16D-06	3.30+04	FTFF
22	3	1.00+00	28	1.7480D+02	6	2	7	1	6.2D+02	1.11D+00	1.D+00	1.D+12	2.D+03	6.45D+00	6.40+03	FFFF
23	10	2.50-01	30	1.7479D+02	6	2	7	1	6.2D+02	1.38D-03	1.D+00	4.D+11	2.D+03	7.40D-05	6.40+03	TFFF
24	1	1.00+00	31	1.7479D+02	6	2	7	1	6.2D+02	7.93D-04	1.D+00	5.D+11	2.D+03	3.81D-06	3.80+05	TTFF
25	3	1.00+00	32	1.7479D+02	5	1	8	2	6.2D+02	5.51D-04	2.D+00	6.D+11	2.D+03	1.19D-07	4.70+04	TTFF
26	2	1.00+00	33	1.7479D+02	6	1	7	2	6.2D+02	6.89D-05	4.D+00	5.D+11	2.D+03	9.43D-08	3.70+04	TTFF
27	2	1.00+00	34	1.7479D+02	5	1	7	3	6.2D+02	1.30D-06	2.D+03	5.D+11	2.D+03	1.12D-07	3.70+04	TTTF
28	1	1.00+00	35	1.7479D+02	5	1	7	3	6.2D+02	1.23D-06	2.D+03	5.D+11	2.D+03	1.93D-07	3.70+04	TTTT

EXIT NP PHASE. INFORM = 0 MAJITS = 28 NFEVAL = 35 NCEVAL = 35

VARIABLE	STATE	VALUE	LOWER BOUND	UPPER BOUND	LAGR MULTIPLIER	RESIDUAL
VARBL 1	FR	.8037732	.1000000	.9000000	.0	.9623D-01
VARBL 2	FR	.8161084	.1000000	.9000000	.0	.8389D-01
VARBL 3	FR	.9000000	.1000000	.9000000	.0	-.1388D-16
VARBL 4	UL	.9000000	.1000000	.9000000	-436.7986	.0
VARBL 5	FR	.9000001	.9000000	1.000000	.0	.1467D-06
VARBL 6	UL	.1000000	.10000000-03	.1000000	-15.19769	.0
VARBL 7	FR	.1078316	.1000000	.9000000	.0	.7032D-02
VARBL 8	FR	.1908367	.1000000	.9000000	.0	.9084D-01
VARBL 9	FR	.1908367	.1000000	.9000000	.0	.9084D-01
VARBL 10	FR	.1908367	.1000000	.9000000	.0	.9084D-01
VARBL 11	FR	505.0452	1.000000	1000.000	.0	495.0
VARBL 12	FR	5.046043	.10000000-05	500.0000	.0	5.046
VARBL 13	FR	72.63782	1.000000	500.0000	.0	71.64
VARBL 14	LL	500.0000	500.0000	1000.000	.6299044	.0
VARBL 15	LL	500.0000	500.0000	1000.000	.2499969	.0
VARBL 16	LL	.10000000-05	.10000000-05	500.0000	.1546715	.0

LINEAR CONSTR	STATE	VALUE	LOWER BOUND	UPPER BOUND	LAGR MULTIPLIER	RESIDUAL
LNCON 1	FR	-499.9991	NONE	.0	.0	500.0
LNCON 2	FR	-.14666590-06	NONE	.0	.0	.1467D-06
LNCON 3	FR	.1387779D-16	NONE	.0	.0	-.1388D-16
LNCON 4	FR	-.83891630-01	NONE	.0	.0	.8389D-01
LNCON 5	FR	-.1233521D-01	NONE	.0	.0	.1234D-01
LNCON 6	UL	-.55511150-16	NONE	.0	-67.20963	.5551D-16
LNCON 7	FR	.55511150-16	NONE	.0	.0	-.5551D-16
LNCON 8	FR	.9999983	NONE	1.000000	.0	.1744D-05

NONLNR CONSTR	STATE	VALUE	LOWER BOUND	UPPER BOUND	LAGR MULTIPLIER	RESIDUAL
NLCON 1	LL	.0	.0	NONE	7.025522	.0
NLCON 2	LL	-.62778180-13	.0	NONE	103.7262	-.6277D-13
NLCON 3	LL	.1127578D-16	.0	NONE	263.3147	.1128D-16
NLCON 4	LL	-.21684040-16	.0	NONE	104.7026	-.2168D-16
NLCON 5	LL	-.15621810-06	.0	NONE	.0	-.1562D-06
NLCON 6	LL	-.11342450-06	.0	NONE	.0	-.1134D-06
NLCON 7	LL	-.18484350-13	.0	NONE	37.48537	-.1848D-13
NLCON 8	LL	-.25951460-14	.0	NONE	402.2756	-.2595D-14
NLCON 9	LL	-.13877790-16	.0	NONE	60.48867	-.1388D-16
NLCON 10	LL	-.20816680-15	.0	NONE	.0	-.2082D-15
NLCON 11	FR	.4924576D-15	.0	NONE	.0	.4925D-15

EXIT NPSOL - OPTIMAL SOLUTION FOUND.

Figure 6. Output from the solution of Dembo 7.

Finally, we wish to emphasize that, despite severe ill-conditioning in the Hessian of the Lagrangian and serious dependencies among the constraints, Dembo 7 is solved in a relatively routine manner. Dependent constraints are successfully omitted from the working set in such a way that its condition estimator never gets much larger than 10^5. Moreover, the final convergence rate, although not superlinear, is quite rapid.

Acknowledgements

The authors would like to thank Robert Burchett of the General Electric Company for providing the 30-bus OPF problem. We are also indebted to Chris Fraley for help in obtaining the results for Dembo 7.

References

Bartholomew-Biggs, M. C. (1982). "Recursive quadratic programming methods for nonlinear constraints", in *Nonlinear Optimization, 1981*, (M. J. D. Powell, ed.), pp. 213–221, Academic Press, London.

Biggs, M. C. (1972a). "Constrained minimization using recursive equality quadratic programming", in *Numerical Methods for Non-Linear Optimization* (F. A. Lootsma, ed.), pp. 411–428, Academic Press, London and New York.

Biggs, M. C. (1972b). Minimization algorithms making use of non-quadratic properties of the objective function, *J. Inst. Maths. Applics.* **8**, pp. 315–327.

Boggs, P. T., Tolle, J. W. and Wang, P. (1982). On the local convergence of quasi-Newton methods for constrained optimization, *SIAM J. Control and Optimization* **20**, pp. 161–171.

Bracken, J. and McCormick, G. P. (1968). *Selected Applications of Nonlinear Programming*, John Wiley and Sons, New York and Toronto.

Brent, R. P. (1973). *Algorithms for Minimization without Derivatives*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Bunch, J. R. and Kaufman, L. C. (1980). A computational method for the indefinite quadratic programming problem, *Linear Algebra and its Applics.* **34**, pp. 341–370.

Coleman, T. F. and Conn, A. R. (1982). Nonlinear programming via an exact penalty function, *Math. Prog.* **24**, pp. 123–161.

Coleman, T. F. and Sorensen, D. C. (1984). A note on the computation of an orthogonal basis for the null space of a matrix, *Math. Prog.* **29**, pp. 234–242.

Dembo, R. S. (1976). A set of geometric test problems and their solutions, *Math. Prog.* **10**, pp. 192–213.

Dennis, J. E., Jr. and Moré, J. J. (1977). Quasi-Newton methods, motivation and theory, *SIAM Review* **19**, pp. 46–89.

Dennis, J. E., Jr. and Schnabel, R. E. (1981). "A new derivation of symmetric positive definite secant updates", *Nonlinear Programming 4* (O. L. Mangasarian, R. R. Meyer and S. M. Robinson, eds.), pp. 167–199, Academic Press, London and New York.

Dennis, J. E., Jr. and Schnabel, R. B. (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Fiacco, A. V. and McCormick, G. P. (1968). *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley and Sons, New York and Toronto.

Fletcher, R. (1981). *Practical Methods of Optimization, Volume 2, Constrained Optimization*, John Wiley and Sons, New York and Toronto.

Fletcher, R. (1983). "Penalty functions", in *Mathematical Programming: The State of the Art*, (A. Bachem, M. Grötschel and B. Korte, eds.), pp. 87–114, Springer-Verlag, Berlin, Heidelberg, New York and Tokyo.

Gabay, D. (1982). Reduced quasi-Newton methods with feasibility improvement for nonlinearly constrained optimization, *Math. Prog. Study* **16**, pp. 18–44.

Gill, P. E., Golub, G. H., Murray, W. and Saunders, M. A. (1974). Methods for modifying matrix factorizations, *Math. Comp.* **28**, pp. 505–535.

Gill, P. E. and Murray, W. (1972). Quasi-Newton methods for unconstrained optimization, *J. Inst. Maths. Applics.* **9**, pp. 91–108.

Gill, P. E. and Murray, W. (1974). Newton-type methods for unconstrained and linearly constrained optimization, *Math. Prog.* **28**, pp. 311–350.

Gill, P. E. and Murray, W. (1978). Numerically stable methods for quadratic programming, *Math. Prog.* **14**, pp. 349–372.

Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. (1984a). User's guide for QPSOL (Version 3.2): a Fortran package for quadratic programming, Report SOL 84-6, Department of Operations Research, Stanford University, California.

Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. (1984b). User's guide for NPSOL (Version 2.1): a Fortran package for nonlinear programming, Report SOL 84-7, Department of Operations Research, Stanford University, California.

Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. (1984c). Procedures for optimization problems with a mixture of bounds and general linear constraints, *ACM Transactions on Mathematical Software* **10**, pp. 282–298.

Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. (1984d). Software and its relationship to methods, Report SOL 84-10, Department of Operations Research, Stanford University, California.

Gill, P. E., Murray, W., Saunders, M. A., Stewart, G. W. and Wright, M. H. (1985a). Properties of a representation of a basis for the null space, Report SOL 85-1, Department of Operations Research, Stanford University, California.

Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. (1985b). The design and implementation of a quadratic programming algorithm, to appear, Department of Operations Research, Stanford University, California.

Gill, P. E., Murray, W. and Wright, M. H. (1981). *Practical Optimization*, Academic Press, London and New York.

Goldfarb, D. and Idnari, A. (1983). A numerically stable dual method for solving strictly convex quadratic programs. *Math. Prog.* **27**, pp. 1–33.

Han, S.-P. (1976). Superlinearly convergent variable metric algorithms for general nonlinear programming problems, *Math. Prog.* **11**, pp. 263–282.

Harris, P. M. J. (1973). Pivot selection methods of the Devex LP code, *Math. Prog.* **5**, pp. 1–28. [Reprinted in *Math. Prog. Study* **4** (1975), pp. 30–57.]

Hock, W. and Schittkowski, K. (1981). Test examples for nonlinear programming, *Lecture Notes in Economics and Mathematical Systems*, Volume 187, Springer Verlag, Berlin, Heidelberg and New York.

McCormick, G. P. (1983). *Nonlinear programming*, Wiley, Wiley-Interscience.

Murray, W. (1971). An algorithm for finding a local minimum of an indefinite quadratic program, Report NAC 1, National Physical Laboratory, England.

Murray, W. and Wright, M. H. (1978). Methods for nonlinearly constrained optimization based on the trajectories of penalty and barrier functions, Report SOL 78 23, Department of Operations Research, Stanford University.

Murray, W. and Wright, M. H. (1982). Computation of the search direction in constrained optimization algorithms, *Math. Prog. Study* **16**, pp. 63–83.

Murtagh, B. A. and Saunders, M. A. (1978). Large-scale linearly constrained optimization, *Math. Prog.* **14**, pp. 41–72.

Nocedal, J. and Overton, M. (1983). Projected Hessian updating algorithms for nonlinearly constrained optimization, Report 95, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, New York.

Powell, M. J. D. (1974). "Introduction to constrained optimization", in *Numerical Methods for Constrained Optimization* (P. E. Gill and W. Murray, eds.), pp. 1–28, Academic Press, London and New York.

Powell, M. J. D. (1977). "A fast algorithm for nonlinearly constrained optimization calculations", in *Numerical Analysis, Dundee, 1977* (G. A. Watson, ed.), pp. 144–157, Springer-Verlag Lecture Notes in Mathematics, Volume 630, Berlin, Heidelberg and New York.

Powell, M. J. D. (1983a). "Variable metric methods for constrained optimization", in *Mathematical Programming: The State of the Art*, (A. Bachem, M. Grötschel and B. Korte, eds.), pp. 288–311, Springer-Verlag, Berlin, Heidelberg, New York and Tokyo.

Powell, M. J. D. (1983b). ZQPCVX a Fortran subroutine for convex quadratic programming, Report DAMTP 83/NA17, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, England.

Robinson, S. M. (1974). Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear programming algorithms, *Math. Prog.* **7**, pp. 1–16.

Schittkowski, K. (1981). The nonlinear programming method of Wilson, Han, and Powell with an augmented Lagrangian type line search function, *Numerische Mathematik* **38**, pp. 83–114.

Schittkowski, K. (1983). On the convergence of a sequential quadratic programming method with an augmented Lagrangian line search function, Math. Operationsforsch. u. Statist., Ser. Optimization **14**, pp. 197–216.

Stewart, G. W. (1973). *Introduction to matrix computations*, Academic Press, London and New York.

Stoer, J. (1984). Foundations of recursive quadratic programming methods for solving nonlinear programs. Paper presented at the NATO Advanced Study Institute on "Computational Mathematical Programming", Bad Windsheim, July 23–August 2, 1984.

Tone, K. (1983). Revisions of constraint approximations in the successive QP method for nonlinear programming, *Math. Prog.* **26**, pp. 144–152.

Wilson, R. B. (1963). *A Simplicial Algorithm for Concave Programming*, Ph. D. Thesis, Harvard University.

Wright, M. H. (1976). Numerical Methods for Nonlinearly Constrained Optimization, Ph. D. Thesis, Stanford University.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: ARO 21592.2-MA	2. GOVT ACCESSION NO.: AD-A155 720
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): MODEL BUILDING AND PRACTICAL ASPECTS OF NONLINEAR PROGRAMMING	5. TYPE OF REPORT & PERIOD COVERED: Technical Report
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Philip E. Gill, Walter Murray, Michael A. Saunders and Margaret H. Wright	8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0267; DAAG29-84-K-0156
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Operations Research - SOL, Stanford University, Stanford, CA 94305	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NR-047-143
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research - Dept. of the Navy, 800 N. Quincy Street, Arlington, VA 22217	12. REPORT DATE: March 1985 13. NUMBER OF PAGES: 39
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): U.S. Army Research Office, P.O. Box 12211, Research Triangle Park, NC 27709	15. SECURITY CLASS. (of this report): UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

sequential quadratic programming
Quasi-Newton Methods
optimization.
numerical methods

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

(See next page)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SOL 85-2: MODEL BUILDING AND PRACTICAL ASPECTS OF NONLINEAR PROGRAMMING
by Philip E. Gill, Walter Murray, Michael A. Saunders and Margaret H. Wright..

Many optimization problems arise from complex models of real-world phenomena. This paper examines the close relationship between certain features of well-posed models and robust optimization methods. First, a list of modelling principles is given, to aid in formulating models suited to solution by modern optimization methods. Quasi-Newton sequential quadratic programming methods for nonlinearly constrained optimization are then discussed. The topics considered include representation and definition of the approximate Hessian of the Lagrangian function; similarities between primal and dual quadratic programming methods; treatment of inconsistent and ill-conditioned subproblems; and properties of various active-set strategies. Finally, the results of solving several test problems are analyzed in detail, including significant characteristics of the overall solution process such as superlinear convergence. Keywords: —) back

END

FILMED

7-85

DTIC